Effect of reducible and irreducible search space representations on adaptive design efficiency: a case study on maximizing packing fraction for solid rocket fuel propellant simulations

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 point inside of a line or a line inside of a rectangle? This is a type of solution degeneracy that often exists physics-based simulations and wetlab experiments, but constraining these degeneracies is often unsupported or difficult-to-implement in many optimization packages, requiring additional time and expertise. So, is the increase in efficiency worth the cost of implementation? We demonstrate that the compactness of a search space (to what extent degenerate solutions and non-solutions are removed) can have a significant effect on Bayesian optimization search efficiency via the Ax platform. As our optimization task, 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. The degeneracies are reflected in the outcomes being identical when: 1. all particle sizes are multiplied by a constant factor, 2. the fractional prevalences of the particle types sum to unity, and 3. sets of log-normal distribution parameters are swapped with each other, respectively. This simulation provides fertile ground for assessing the impact of multiple constraints on search efficiency, with a total of eight search space types which ranges from none up to all three constraints imposed simultaneously. Contrary to intuition, we find that, on average, the most compact search space performs worse than the least compact search space ((0.692 ± 0.036) vs. (0.699 ± 0.016), respectively) over 50 iterations due to the interactions of the non-linear size-invariance degeneracy with other degeneracy types. The most efficient search space in terms of both predicted and validated outcomes is the combination of the composition and permutation constraints, resulting in a mean packing fraction of (0.728 ± 0.010) over 50 iterations, where randomly sampled volume fractions are typically no less than 0.6. We recommend that optimization practitioners in the physical sciences carefully consider the impact of removing search space degeneracies on search efficiency prior to running expensive optimization campaigns.

Keywords: Bayesian optimization, sequential learning, adaptive design, active learning, concurrency, AxSearch, RayTune, Ax platform, Python, optimization, constrained optimization

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 [2–8] 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, 7] or tuning hyperparameters of a surrogate model [3, 10] among other applications such as experimental discovery [6–8] and crystal structure prediction [11–14].

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. [15] 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. [15], Hickman et al. [16] observed the effect of model choice and task on single- and multi-objective search efficiency under a single constraint. They 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. Here, we focus on a single model (Gaussian process expected improvement (GPEI) via the Adaptive Experimentation platform) 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 question:

> How does creating an irreducible representation for an adaptive design search space affect search efficiency for small datasets?

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

[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" [9].

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 [17].

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 [18], 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" [19], 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 small 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 [18] 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 Section S4 for discussion and figures 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

2.1. Adaptive Experimentation Platform and RayTune

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" [10] and refer to this as Ax. In prior work [10], 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 choose to use the default Bayesian model, GPEI, instead of SAASBO for several reason as follows:

1. Number of cores is limited in this study, and SAASBO is computationally expensive; use of concurrency results in trial evaluation and candidate generation often occurring simultaneously

2. There are fewer dimensions in this study and SAASBO's exceptional performance tends to manifest with high-dimensional problems where certain features are much less important than others[3]

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

[3] Nonetheless, it would be interesting to see if SAASBO is capable of quickly resolving the degenerate structure of certain search spaces.

3. Basic and preliminary results (not shown) suggest that the performance between SAASBO and GPEI may be on par with each other for this problem; however, this could be verified much more thoroughly

We refer to the GPEI implementation within the Ax platform as `GPEI`.[4]

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 for the Bayesian optimization trials. 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[5] 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[6] 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 Ray-Tune Trial Schedulers documentation. These types of scheduling algorithms can be applied to any combination of offline/online[7] 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).

2.2. Validation

Each optimization is repeated 5 times (each using a unique, fixed seed for the random number generator) with the fixed design budget and setup as described in Section 2.1. Leave-one-out cross-validation is performed for each final optimization dataset, and the best in-sample predictions[8] are repeated 50 times, for which the mean and standard deviation are calculated. 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; in other words, this validation step is central to the findings of this study.

[4] 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.

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

[6] 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. [20] for two types of conditioning: sequential greedy approaches and "fantasy" models. See also Wilson et al. [21].

[7] 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.

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

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.

2.3. Particle Packing Simulations

A proprietary Windows executable was used. The driver of these is a Python class. 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.

Theoretical details of the particle packing simulations are given in Davis and Carter [22], Webb and Davis [23], for which a summary is provided in the second paragraph of the Motivation section in Hall et al. [18]. In short, these simulations involve dropping particles sampled from a predefined distribution of particle sizes inside of a cylinder at randomized locations [18].

The datasets in Hall et al. [18] 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. [18] 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. [18] due to the differences in distribution sampling. For a comparison of Hall et al. [18] 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.

2.4. 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. [24] found that mapping symmetrically related sets of parameters to an irreducible representation (i.e. a fundamental zone in crystallographic terms[9]) exhibited distinct advantages related to accuracy and computational efficiency of distance calculations[10] 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) [25, 26] (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 $0.4 + 0.6 = 1.0$) [27–38] (referred to as "comp"), or a set of parameters that exhibits permutation invariance (e.g. $Al_2O_3 \equiv O_3Al_2$) [24, 28] (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 [39] which is not addressed in this work. When no additional parameter constraints other

[9] 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)

[10] 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

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 particles.

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

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 [40] or t-distributed stochastic neighbor embeddings [41] in that only redundant information is lost[11] 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 S9.

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 degenerate search spaces are given in Figure 1.

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

Consistent with the validation results from Figure 2, the comp/order constrained configuration (Figure 3f) exhibits the best predicted (optimized) volume fraction out of eight search spaces. Likewise, the predicted (Figure 2) and validated (Figure 3g) outcomes for the size/order constrained configuration mirror each other as the worst out of the eight search spaces. In other words, the best in-sample predictions appear to match the validated outcomes, despite an optimization budget of 50 iterations over 7-9 tunable parameters. This is a reassuring finding, especially if it applies to situations where the validation is much more expensive or infeasible to perform, where someone must "trust" the best in-sample predictions despite it using a model trained on noisy data.

The authors think it likely that use of data augmentation as described in Appendix A.3 would result in better performance than the order constraint; 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 S6.

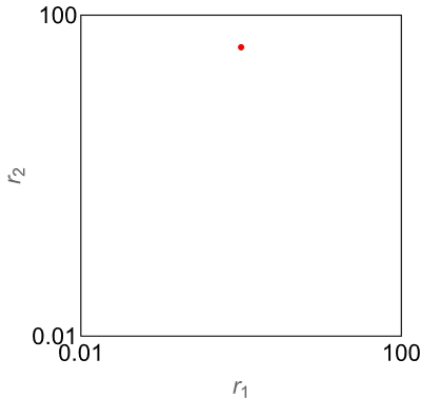
3.2. Interpretable Model Characteristics

In this subsection, we probe some interpretable characteristics of the `GPEI` model through particle size distribution visualizations (Section 3.2.1), feature importances (Section 3.2.2), and 2D contours for two of the compositional variables (Section 3.2.3). Plots with additional features and details are given in Sections S4, S5 and S8, respectively. Additionally, leave-one-out cross-validation results are provided in Section S7.

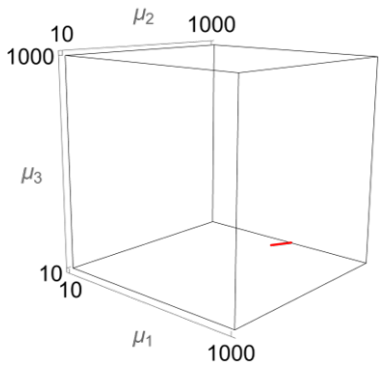
3.2.1. Solutions visualized as summed distributions

Bounds-only exhibits a variety of summed distribution shapes across the seeded runs. Composition exhibits greater similarity with two primary distributions, one small and one large, and mild variation in the peak widths and heights. Size also exhibits some homogeneity across the seeded runs,

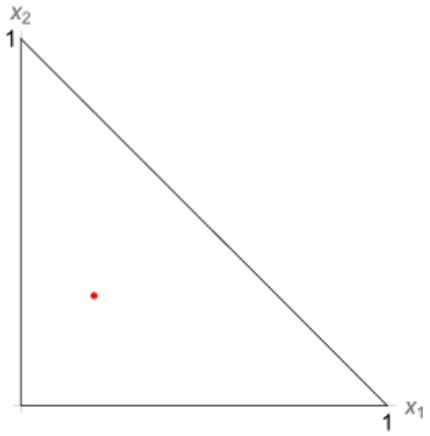
[11] "Only redundant information is lost" assumes that the constraints imposed are consistent with the actual behavior of the objective function.



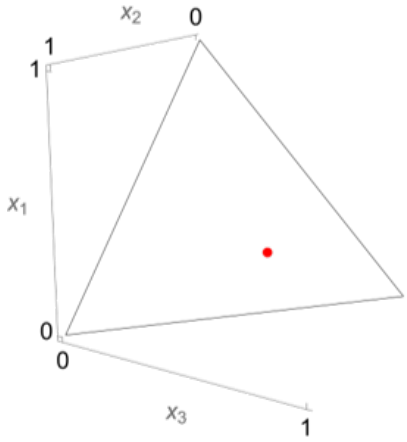
(a) $r_1 = \frac{\tilde{x}_1}{\tilde{x}_3}, r_2 = \frac{\tilde{x}_2}{\tilde{x}_3}$



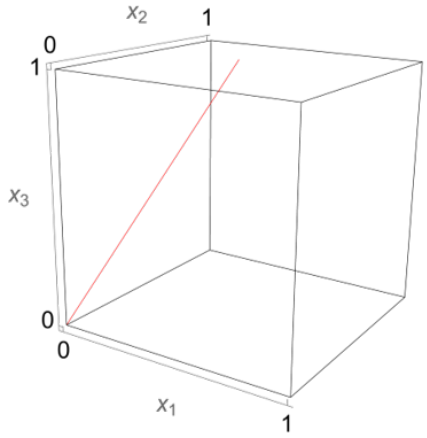
(b) Bound constraints only



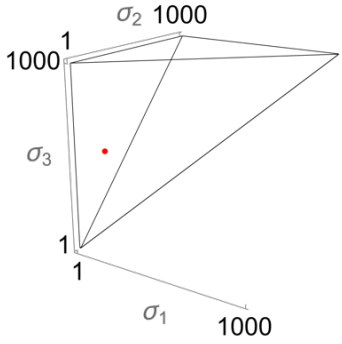
(c) $p_1 + p_2 \leq 1.0$



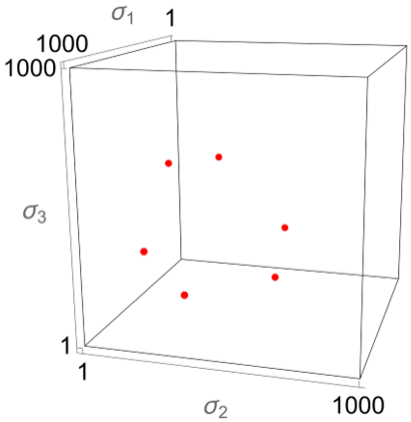
(d) $p_1 + p_2 + p_3 = 1.0$



(e) Bound constraints only



(f) $s_1 < s_2 < s_3$



(g) Bound constraints only

Figure 1: Irreducible and reducible search spaces for size, compositional, and permutation constraints.

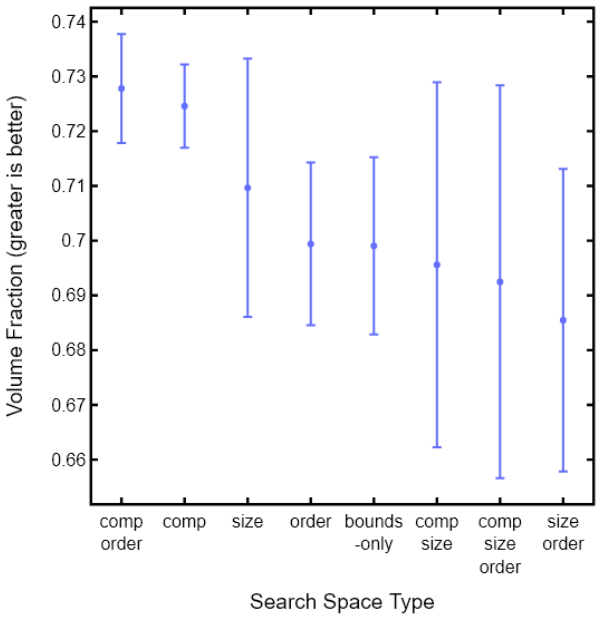


Figure 2: Packing fraction mean and standard deviation of validated results using 50 repeat runs for each of the 8 search space types, sorted by decreasing mean packing fraction.

with large bounds (and therefore maximum particle size ratios) and peaks that appear sharp due to the widened x-axis bounds. Order is not very homogeneous across seeded runs, exhibiting both unimodal, bimodal, and trimodal summed distributions. Comp/Size is likewise varied. Interestingly, it seems to exhibit aspects from both the Comp-only and Size-only solutions. There are a mixture of sharp/broad peaks and small/large particle size bounds. Comp/Order is fairly homogeneous across seeded runs, often producing jagged peaks in-between the smallest and largest distributions due to summation of truncated distributions whose bounds overlap near peak locations. Additionally, we note that Comp/Order had the best predicted (Figure 3) and validated (Figure 2) performance out of the eight search space types. Size/Order is varied and exhibits a mix of characteristics from the Size-only and Order-only solutions: large particle-size bounds/sharp peaks and peaks with similar heights[12], respectively. We note that Size/Order had the worst performance out of the eight search spaces. Finally, Comp/Size/Order (All) exhibits large bounds and is quite varied in terms of shape, and the solutions exhibit similarity with the Size/Order solutions.

In all cases involving Size, instances of much higher upper bounds on the particle size are observed (up to approx. 200), and in all cases not involving Size, the particle size upper bound is restricted to less than approx. 20. This seems significant as the Size reparameterization in the presence of other constraints always had a deleterious effect on the predicted and validated average performance.

Distribution visualizations on a per-seed basis with additional plot features and annotations are given for seeds 10, 11, 12, 13, and 14 are given in Section S4.

3.2.2. Feature Importances

Average feature importances[13] across five seeded runs are given in Figure 5. We note that each search space is characterized by different sets of features, ranging from seven to nine total features.

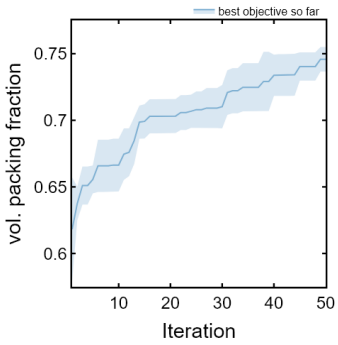
One of the characteristics that stands out is the large standard deviations for many of the features. In other words, separate optimization runs did not necessarily assign the same features as being most important regardless of the search space.

We would have expected that for the search space with the best predicted and validated outcomes (Composition reparameterization + order constraint), the feature importances would have tighter standard deviations than others; however, this does not appear to be the case. The search space with the worst average performance (size reparameterization and order constraint) exhibits fairly comparable standard deviations for the search spaces, albeit for different parameters.

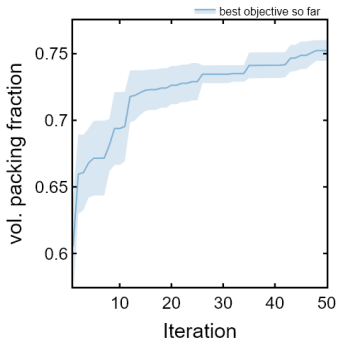
Individual feature importances based on the fitted, inverse lengthscales of the anisotropic Gaussian kernels for each of the seeded runs are given in Section S5.

[12]Size/Order shows peaks with similar heights as opposed to the first-peak-low, second-peak-high trend.

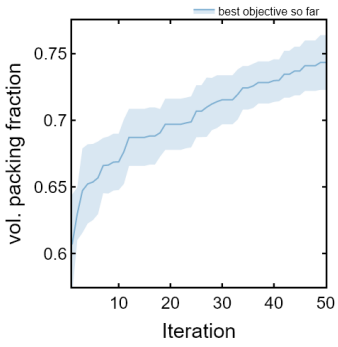
[13]Feature importances are based on based on the inverse lengthscales of an anisotropic Gaussian kernel.



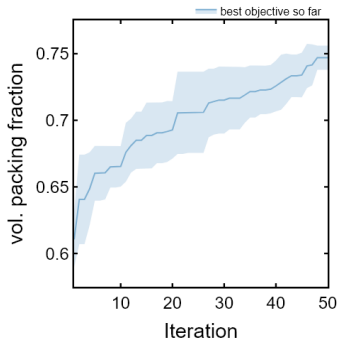
(a) Bound constraints only



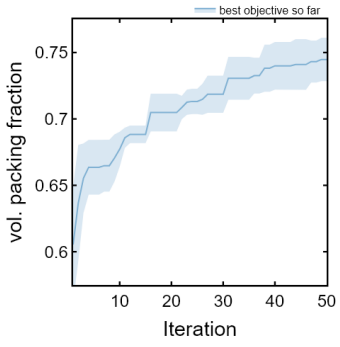
(b) Composition reparameterization



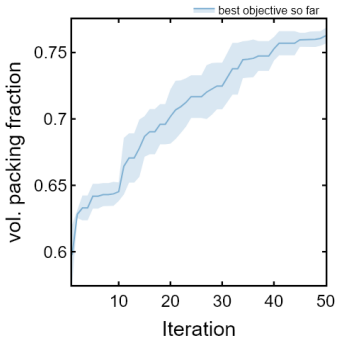
(c) Size reparameterization



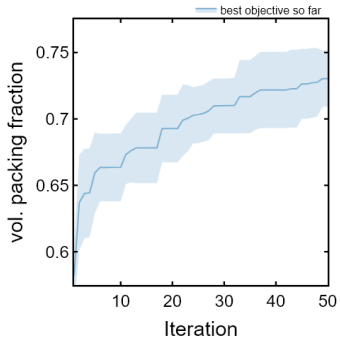
(d) Order constraint



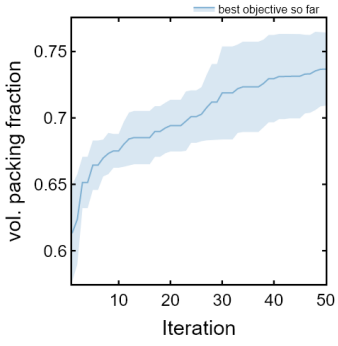
(e) Composition and size reparameterizations



(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. By default, Ax uses the the best in-sample predictions rather than the noisy measured values unless a reasonable fit is not obtained.

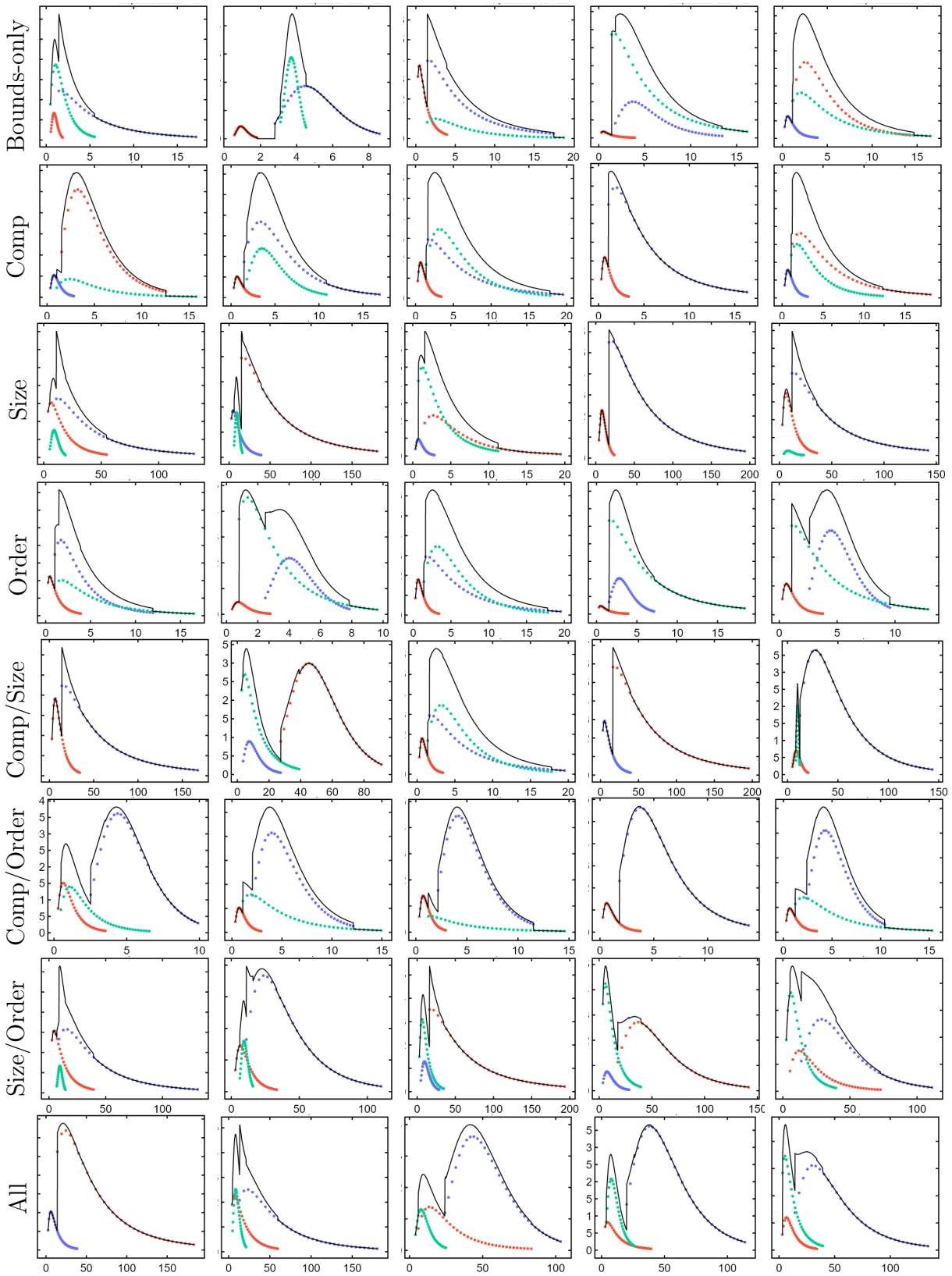
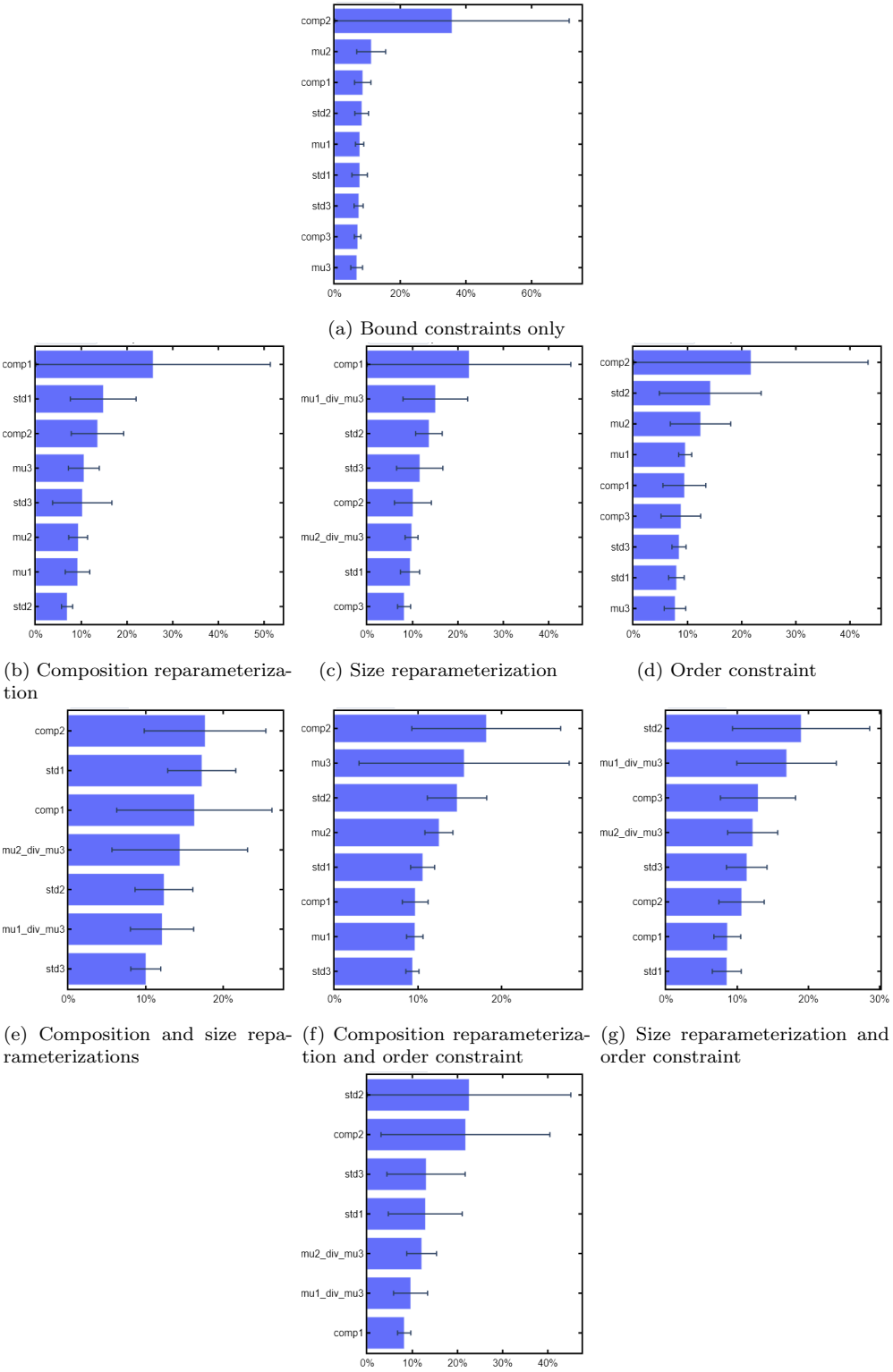


Figure 4: Summed distributions (black) and each of the component distributions (red, blue, and optionally green) for mass fraction vs. particle size for the eight search spaces (rows) repeated for five seeded runs (columns).



(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 5: Average feature importances for the eight search spaces across five seeded optimization runs with standard deviations as error bars.

3.2.3. 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 five seeded optimization runs are given in Figure 4.

We observe qualitative differences between the various search spaces. Of the eight search spaces, only Comp/Order appears to have solutions for `comp1` and `comp2` that very close to each other across each of the optimization runs. Interestingly, the Comp-only and Order-only constraints do not exhibit this homogeneity of search solution, indicating that each of these constraints has a synergistic effect on the ability to "compact" the search space. This is further corroborated by the fact that Comp/Order is the highest performing search space both in terms of predicted and validated outcomes.

The other seven search spaces exhibit a local optimum in differing locations, and in some cases multiple local optima.

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 S8.

4. Future Work

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)?
- To what extent can multi-fidelity optimization reduce total search cost?
- 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. genetic algorithms, random forest based BO [42])?
- 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?
- Could replacing the Bayesian model with SAASBO help in quickly resolving the degenerate structure of reducible search spaces?
- 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?

5. Conclusion

Contrary to intuition, the most compact search space is not always the most efficient from a Bayesian optimization perspective. In particular, the application of a constraint that introduces a non-linear transform tends to have a deleterious effect on average search efficiency, but only when in the presence of the other constraints. In other words, removing search space degeneracy does not always equate to more efficient optimization. The performance was worse than the least compact search space in all three cases where the non-linear transform was combined with at least one other constraint. Consistent with expectation, there were compacted search spaces that exhibited better performance than the least compact space. We show that careful consideration and application of linear and order constraints dramatically increases the search efficiency relative to less-constrained search spaces for a physics-based simulation with seven

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

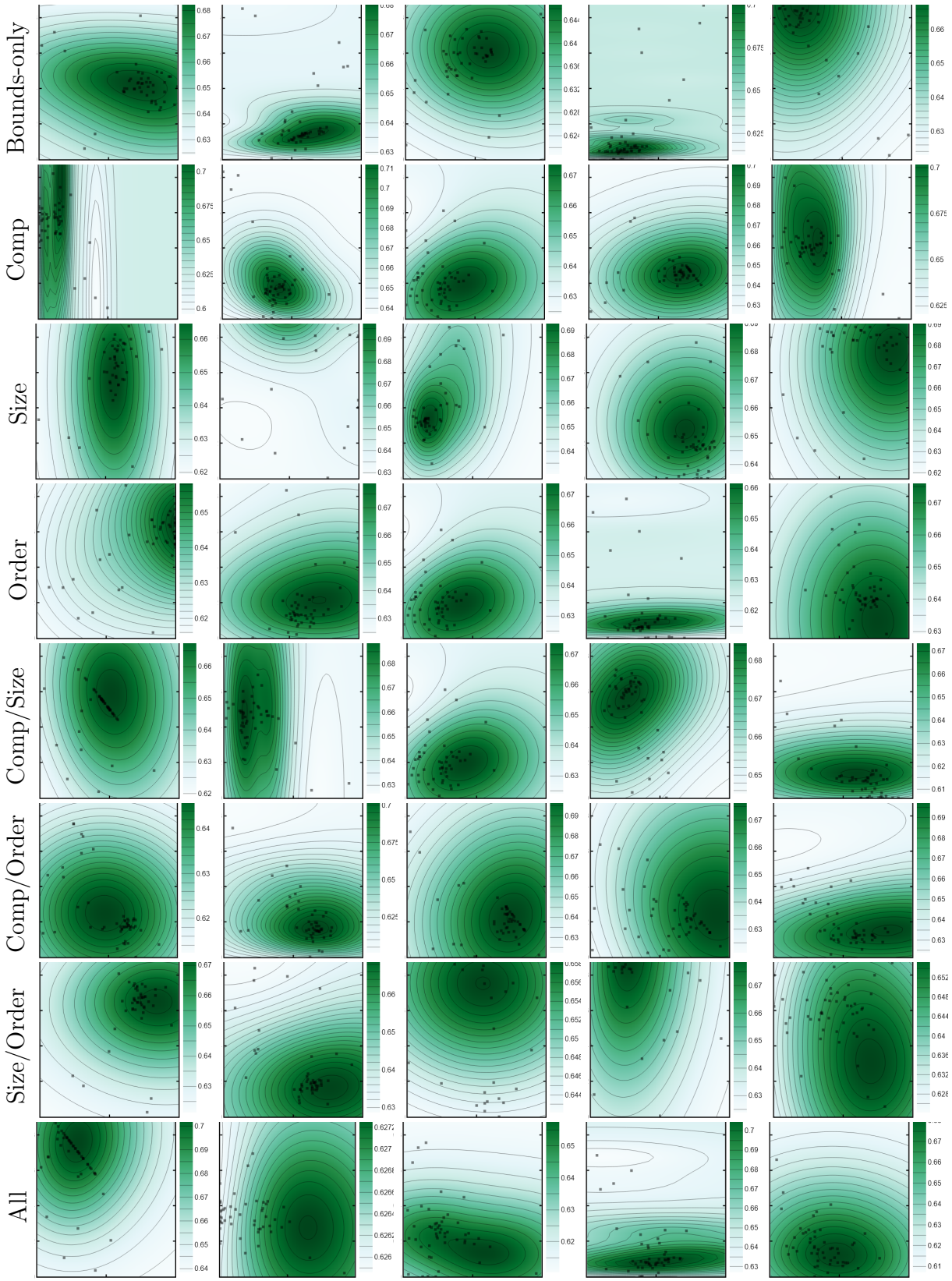


Figure 6: 2D contour plots of `comp2` (y-axis) vs. `comp1` (x-axis) for each of the eight search spaces (rows) and each of the five seeded optimization runs.

to nine tunable parameters depending on the compactness of the search space. A linear equality constraint and an order constraint acted synergistically to produce the best predicted and validated outcomes. This was also manifest in visualizations of the solutions across repeated optimization runs and 2D contour characteristics of a subspace of the full search spaces. Average 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 nonlinear constraints or reparameterizations on their search spaces, especially when expensive physics-based simulations or wetlab experiments 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.

Appendix A Reparameterizations and Constraints

A.1 Size Invariance

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

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

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):

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

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

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. This can be reparameterized, albeit with some distortion of the original search space, as a linear inequality constraint 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.

A.3 Permutation Invariance

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

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

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.

Glossary

BO Bayesian optimization 2, 3, 12, 15

GPEI Gaussian process expected improvement 2–4

SAASBO sparse axis-aligned subspaces Bayesian optimization 3, 4, 12

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. Plots were produced via Plotly [43] and Ax's plotting wrapper functions. Several tables were formatted via an online formatter [44] and the auto-paper methodology [45] 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.

CRediT Statement

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

Data Availability

There is no raw data associated with this study. The processed data required to reproduce these findings is available to download from `https://gi`

thub.com/sparks-baird/bayes-opt-particle-packing as v0.1.0 [].

The code required to reproduce these findings is hosted at https://github.com/sparks-baird/bayes-opt-particle-packing as v0.1.0 [].

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. 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.

[7] 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.

[8] 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.

[9] M. Kuhn, Acquisition functions, https://tune.tidymodels.org/articles/acquisition_funct ????

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] T. Yamashita, Hybrid algorithm of Bayesian optimization and evolutionary algorithm in crystal structure prediction (????) 21.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

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

[39] 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.

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

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

[42] 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.

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

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

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