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: Supporting Information

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

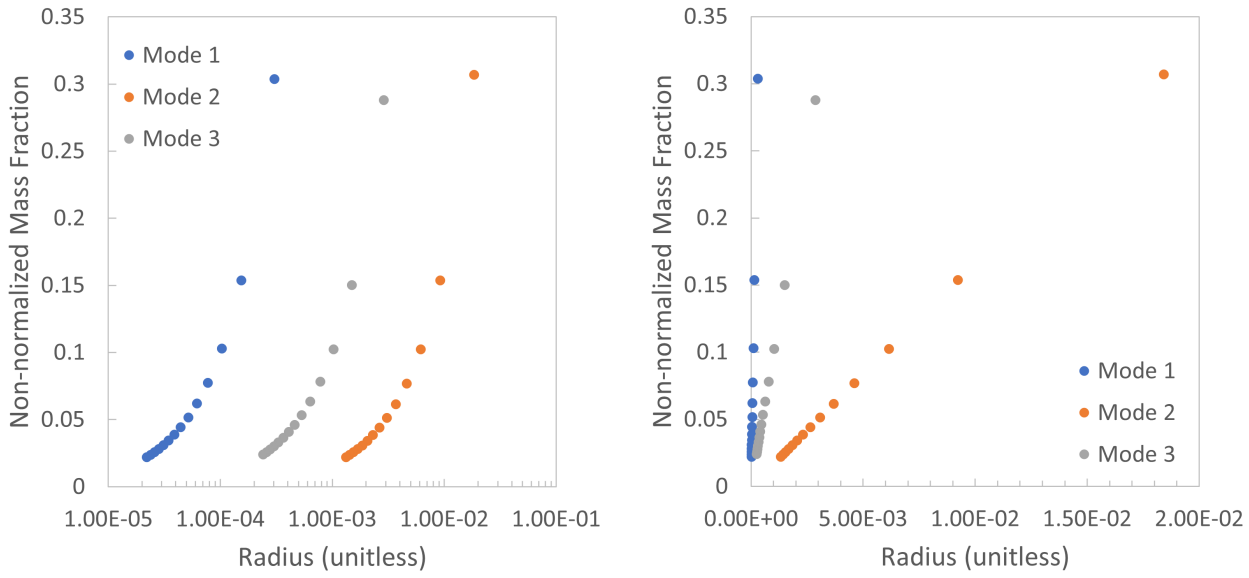
[a]*Department of Materials Science and Engineering, University of Utah, Salt Lake City, UT 84108, USA*
[b]*Northrop Grumman Innovation Systems, 9160 UT-83, Corinne, UT 84307*

Contents

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

S1. Approximating Particle Size Distributions as Log-normal Distributions

Legacy datasets used a positive linear relationship between mass fraction and particle size (Figure S1). The actual data that was produced in Hall et al. [1] is opposite to what is described in Fig. 2 of Hall et al. [1] due to an error in how the distributions were processed in internal MATLAB scripts (not provided in the associated repository https://github.com/MrJasonBear/Particlepacking).



(a) Radius logarithmically-scaled

(b) Radius linearly-scaled

Figure S1: Legacy data which exhibits a linear trend between mass fraction and particle radius, with particle radius logarithmically (a) and linearly (b) scaled for comparison with Hall et al. [1] Fig 2. and to emphasize the linear trend, respectively.

In this work, we create particle size distributions more representative[1] of milling[2–4], consistent with typical solid propellant manufacturing techniques, by representing these as log-normal distributions. We use the standard deviation as the shape parameter directly and a scaled version of mean such that if one were to take the mean of an infinite number of samples of the distribution, this would reflect the input mean specified:

```
s_mode_low, s_mode_upp = lognorm.ppf(alphas, s, scale=scale)
s_mode_radii = np.linspace(s_mode_low, s_mode_upp, running_size)
```

where `alphas` are the lower and upper quantiles used to truncate the distribution and are chosen dynamically in a way that limits the total number of distinct submodes (i.e. particles with a given size and mass fraction) to fewer than 100 across all modes (i.e. particle distributions). This is to comply with a software limitation of ParPack, the proprietary particle packing simulation software, which only processes up to a certain number of submodes.

In order to bound the computational complexity of the particle packing simulations, we additional restrict the lower and upper limits of a sampled distribution to 0.25 and 4 times the `scale` of the distribution, respectively.

[1] While this may be more representative, experimentally accessible log-normal distribution parameters of real particle size distributions is a more complicated matter and depends on processing conditions. The authors are not aware of any authoritative sources on reasonable experimental bounds (or accessible combinations) of log-normal distributions based on milling techniques.

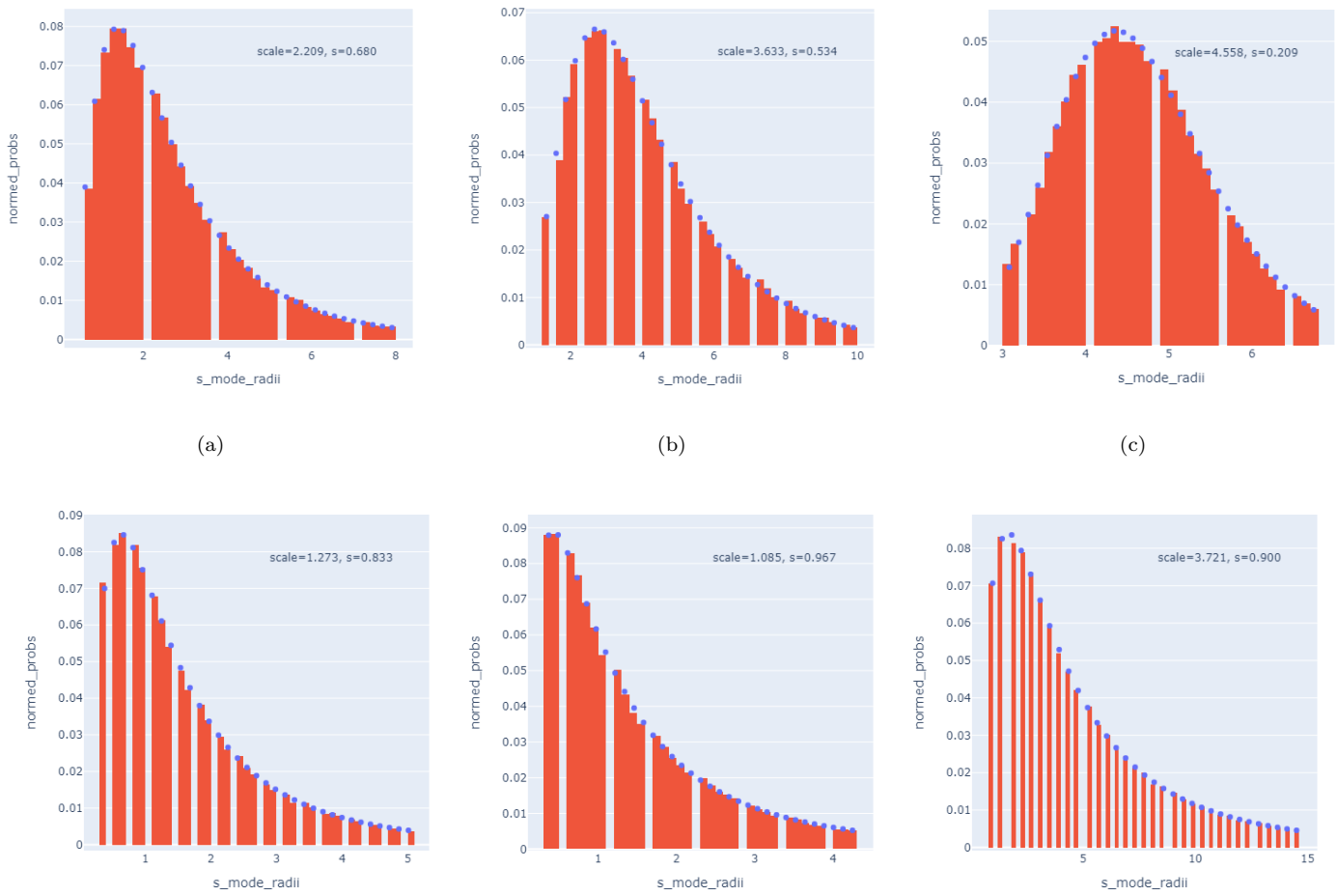


Figure S2: Examples of discrete truncated log-normal distributions, including the particle mass fractions and radii supplied to ParPack, overlaid with histograms of 100 000 points that were sampled from the discrete, truncated distribution using `random.choice` with weights. Scale (`scale`) and shape (`s`) parameters are given as text annotations.

```
    max_ratio = 16
    upp = np.sqrt(max_ratio)  # e.g. 4
    low = 1 / upp  # e.g. 0.25
    s_mode_radii = s_mode_radii[
        np.all(
            [s_mode_radii > low * scale, s_mode_radii < upp * scale],
            axis=0,
        )
    ]
```

Examples of max-ratio sampled distributions are given in Figure S2. Additionally, a gridded sampling of the `scale` and `shape` parameters used in this work are given in Figure S3 and replot the gridded sampling with a logarithmically scaled x-axis to accentuate differences between the parameter combinations Figure S4.

Details of the Python implementation can be found in https://github.com/sparks-baird/bayes-opt-particle-packing/blob/main/boppf/utils/particle_packing.py.

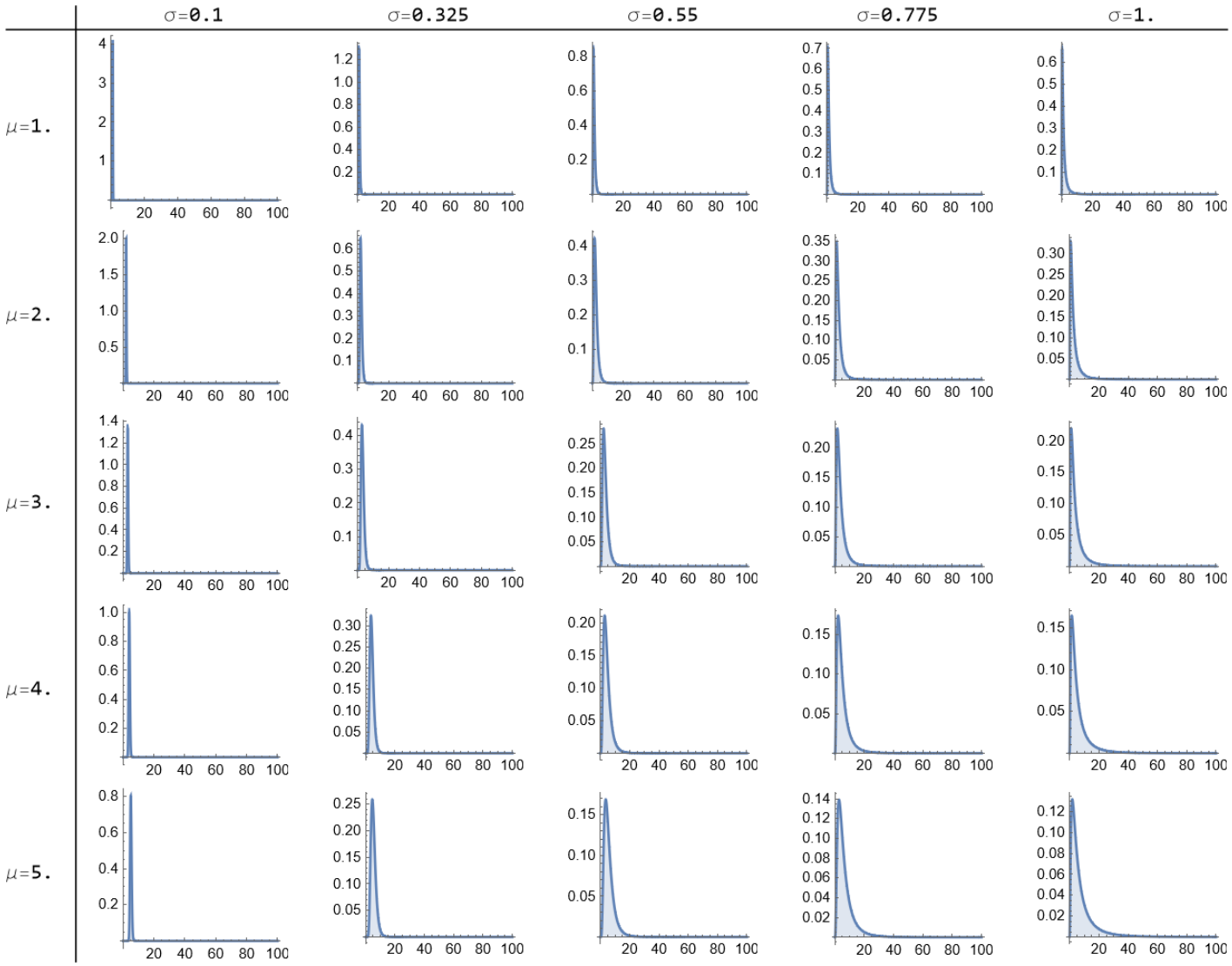


Figure S3: A gridded sampling of mean (`scale`) and standard deviation (`s`) for the bounds described in Table 1 with equal, linear scaling of the x-axis. Plots were produced via Mathematica with the distributions defined by `LogNormalDistribution[Log@mu, sigma]`.

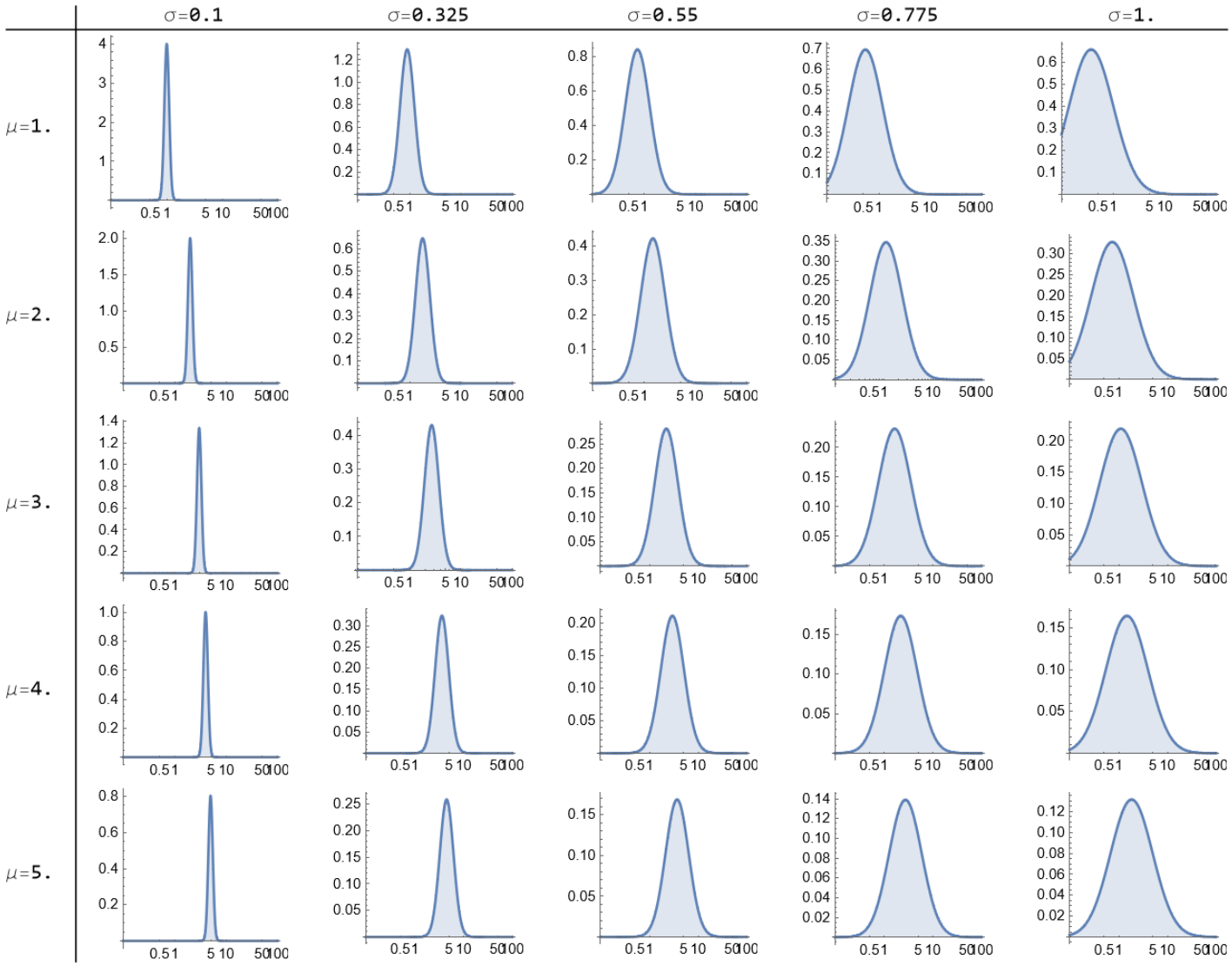


Figure S4: A gridded sampling of mean (`scale`) and standard deviation (`s`) for the bounds described in Table 1 with equal, logarithmic scaling of the x-axis. Plots were produced via Mathematica with the distributions defined by `LogNormalDistribution[Log@mu, sigma]`.

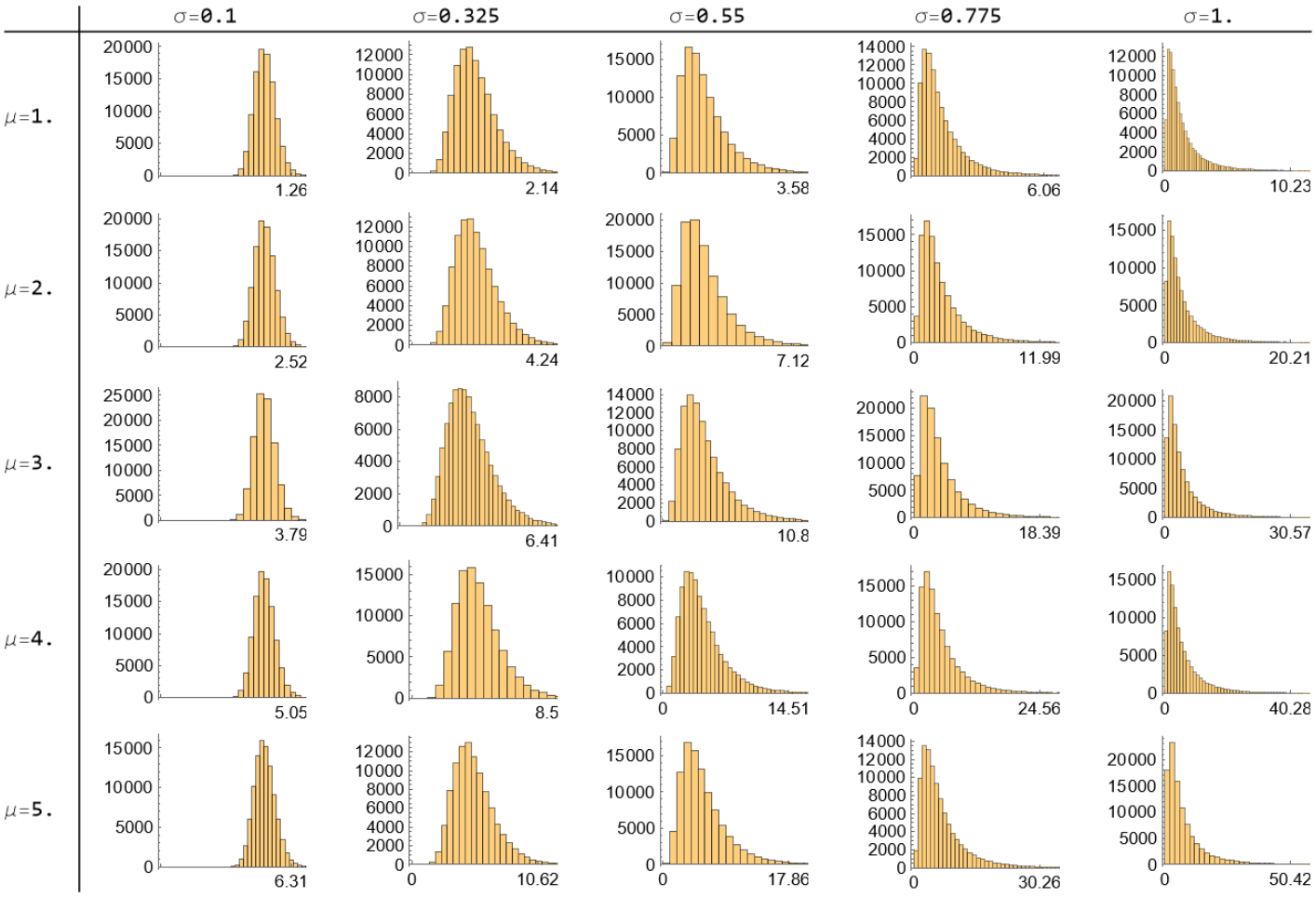


Figure S5: Histograms of 100 000 points sampled from log-normal distributions defined by a gridded sampling of mean (`scale`) and standard deviation (`s`) for the bounds described in Table 1 with unequal, linear scaling of the x-axis. Plots were produced via Mathematica with the distributions defined by `LogNormalDistribution[Log@mu, sigma]`.

S2. Size Invariance

Resampled distributions (non-uniformly scaled x-axes) are given in Figure S5 to show the similarity of distribution shape when changing `scale` for a fixed `shape`. When two distributions are overlapping Figure S6, the shape similarity trend also holds, justifying the notion that our use of size reparameterization in this work does not change the accessible parameterizations, only rather the representation of those parameterizations. In other words, this suggests that after size reparameterization, the same solutions (albeit with less degeneracy) exist in the reduced search space and no new solutions arise from the reparameterization.

S3. Simulation variation

Packing fraction vs. particle size showing run-to-run variation for the legacy distribution shown in Figure S1 is given in Figure S7. Run-to-run volume fraction variations for the truncated log-normal distributions used in this work are not provided.

S3.1. Solutions visualized as summed distributions

The solutions visualized as summed distributions are given in Figure S9. For brevity, only the first 5 out of 10 seeded campaigns are shown.

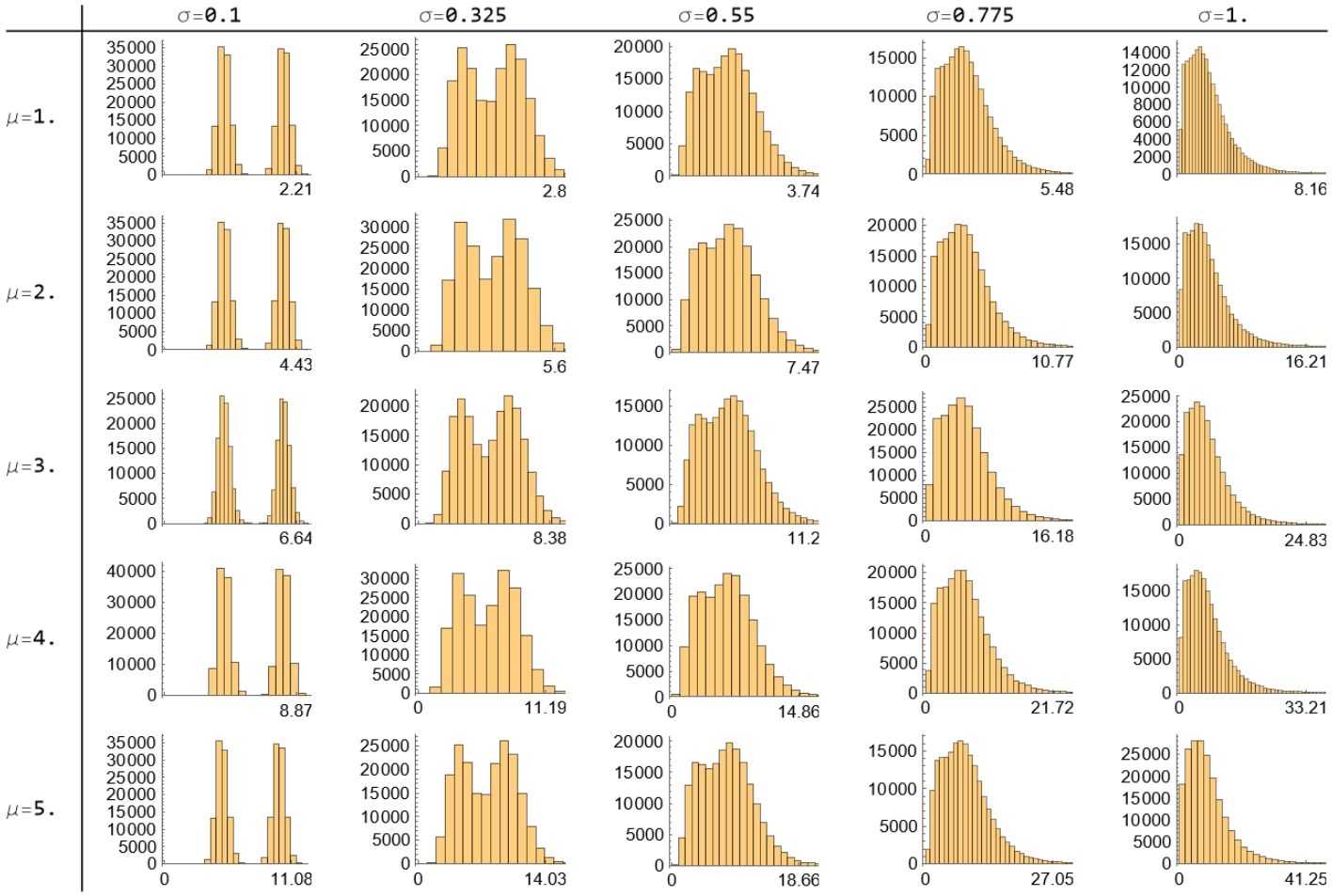


Figure S6: Histograms of 200 000 points (100 000 per distribution) sampled from two log-normal distributions. The first log-normal distribution is defined by a gridded sampling of mean (`scale`) and standard deviation (`s`) for the bounds described in Table 1 with unequal, linear scaling of the x-axis. The second distribution is fixed at twice the mean and half the standard deviation of the first distribution. Plots were produced via Mathematica with the distributions defined by `LogNormalDistribution[Log@mu, sigma]`.

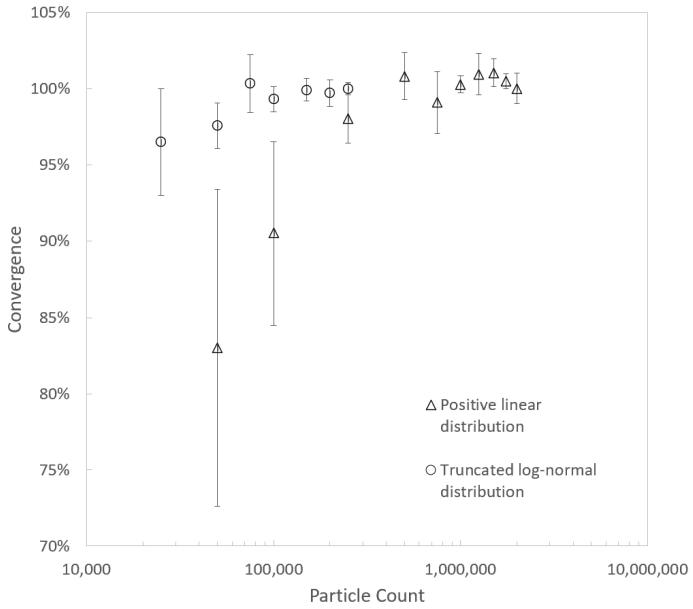
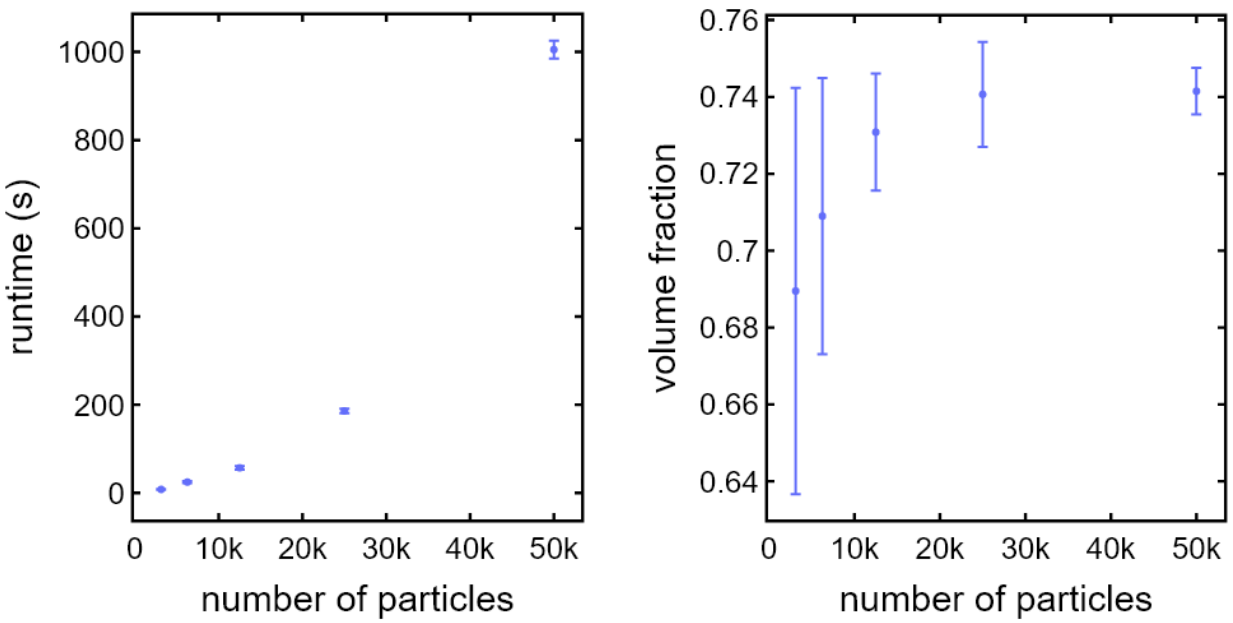


Figure S7: Convergence of packing fraction vs. particle size showing run-to-run variation for a legacy distribution shown in Figure S1 and a truncated, discrete, log-normal distribution. Five repeats were used for each, except for 1.75×10^6 particles for the legacy distribution, which had 4 repeats.



(a) (b)

Figure S8: Average runtime (a) and volume fractions (b) vs. number of particles of six repeats with standard deviations as error bars. The following parameters were used: `scales = [4.999999999, 1, 3.195027437]; shapes =[0.541655268 , 0.705662075, 0.755319764]; fractions =[0.840443899, 0.159556101, 0.0]`.

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 S3.2.

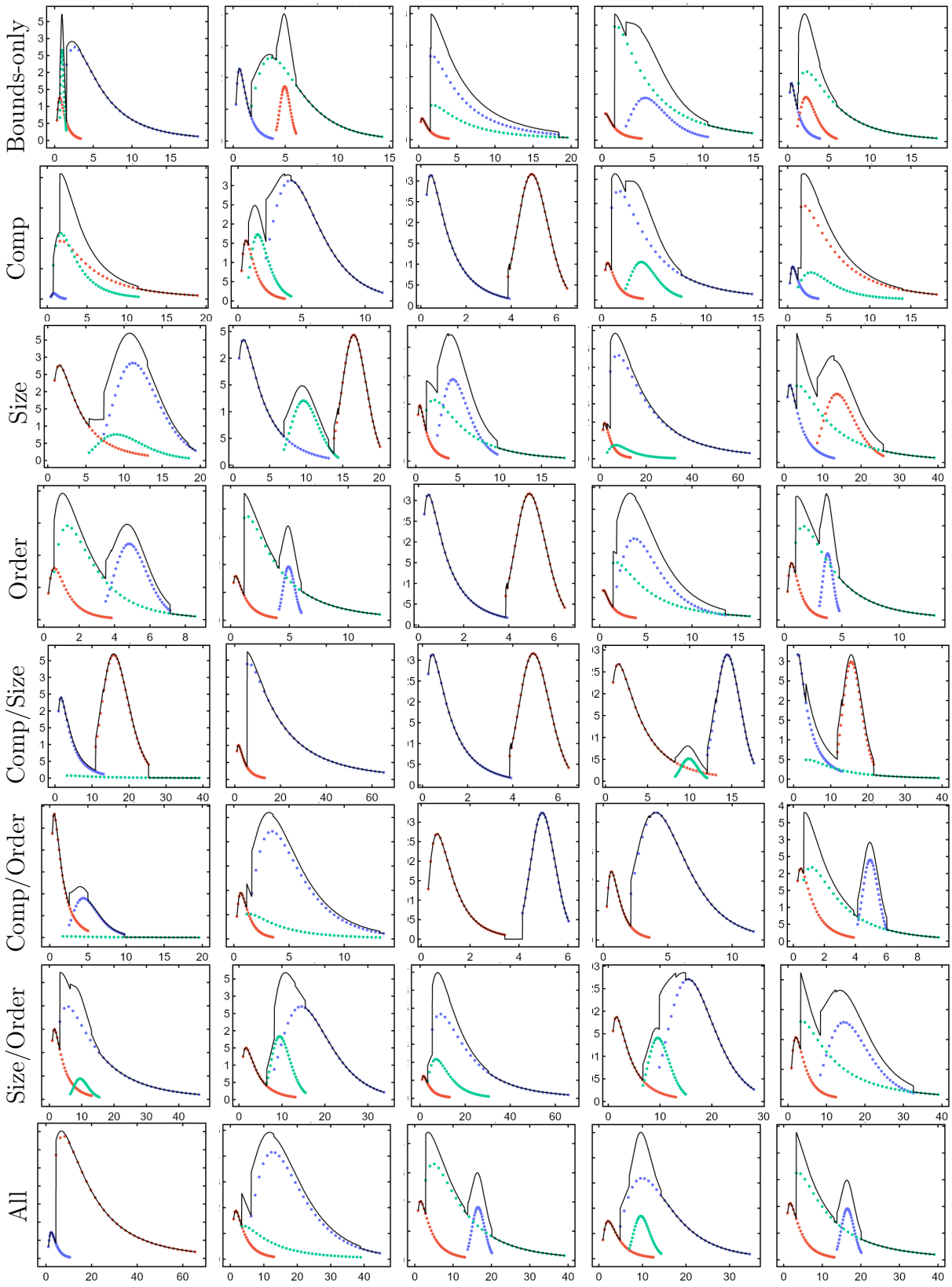
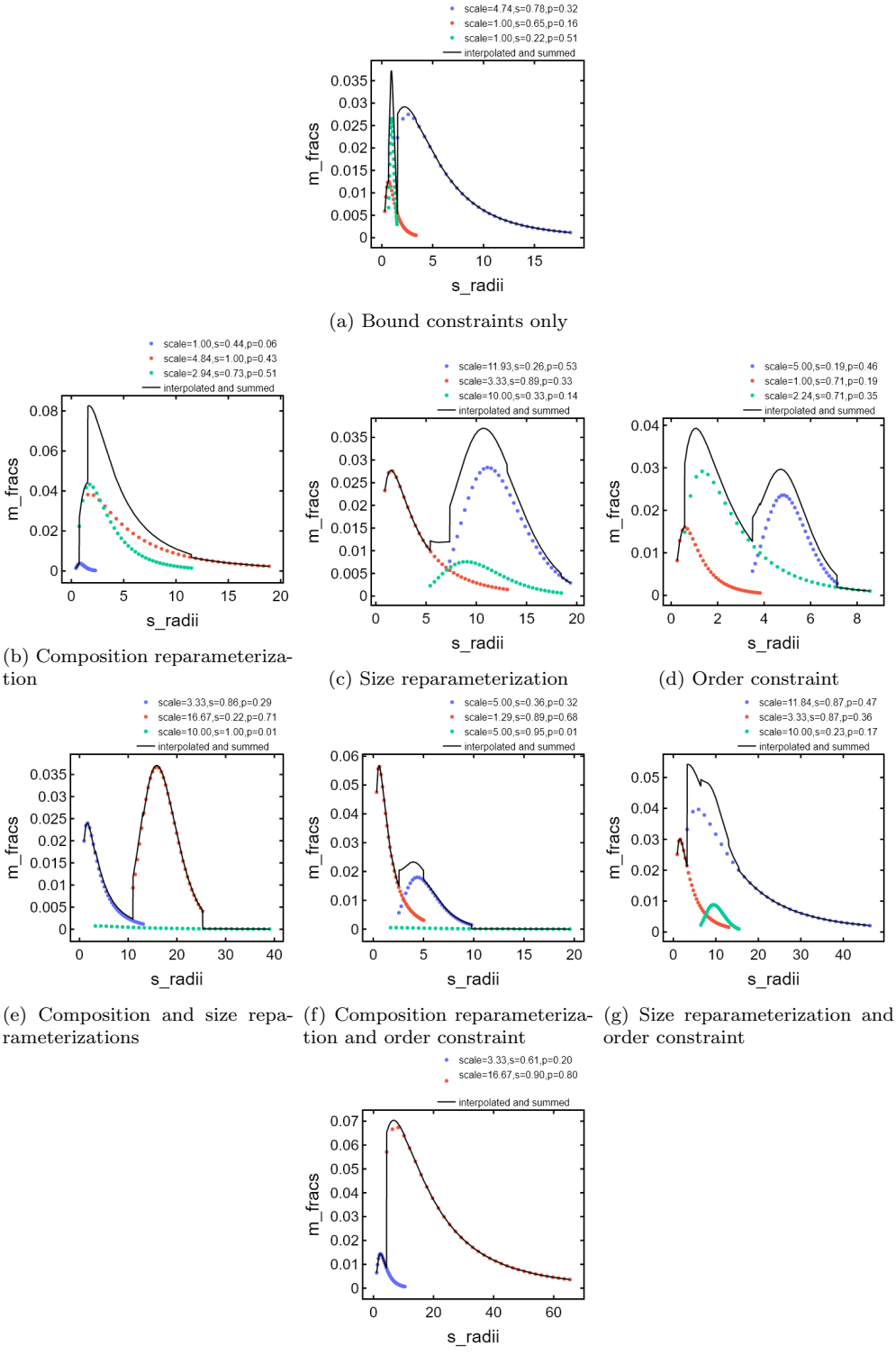


Figure S9: Summed distributions (black) and each of the component distributions (red, blue, and optionally green) for mass fraction vs. particle size for the eight search spaces (rows) repeated for five seeded runs (columns) using Gaussian process expected improvement.

S3.2. Solutions visualized as summed distributions on a per-seed basis

Distribution visualizations for seeds 10, 11, 12, 13, and 14 are given in Figures S10–S14.

S3.3. Seed=10



(a) Bound constraints only

(b) Composition reparameterization

(c) Size reparameterization

(d) Order constraint

(e) Composition and size reparameterizations

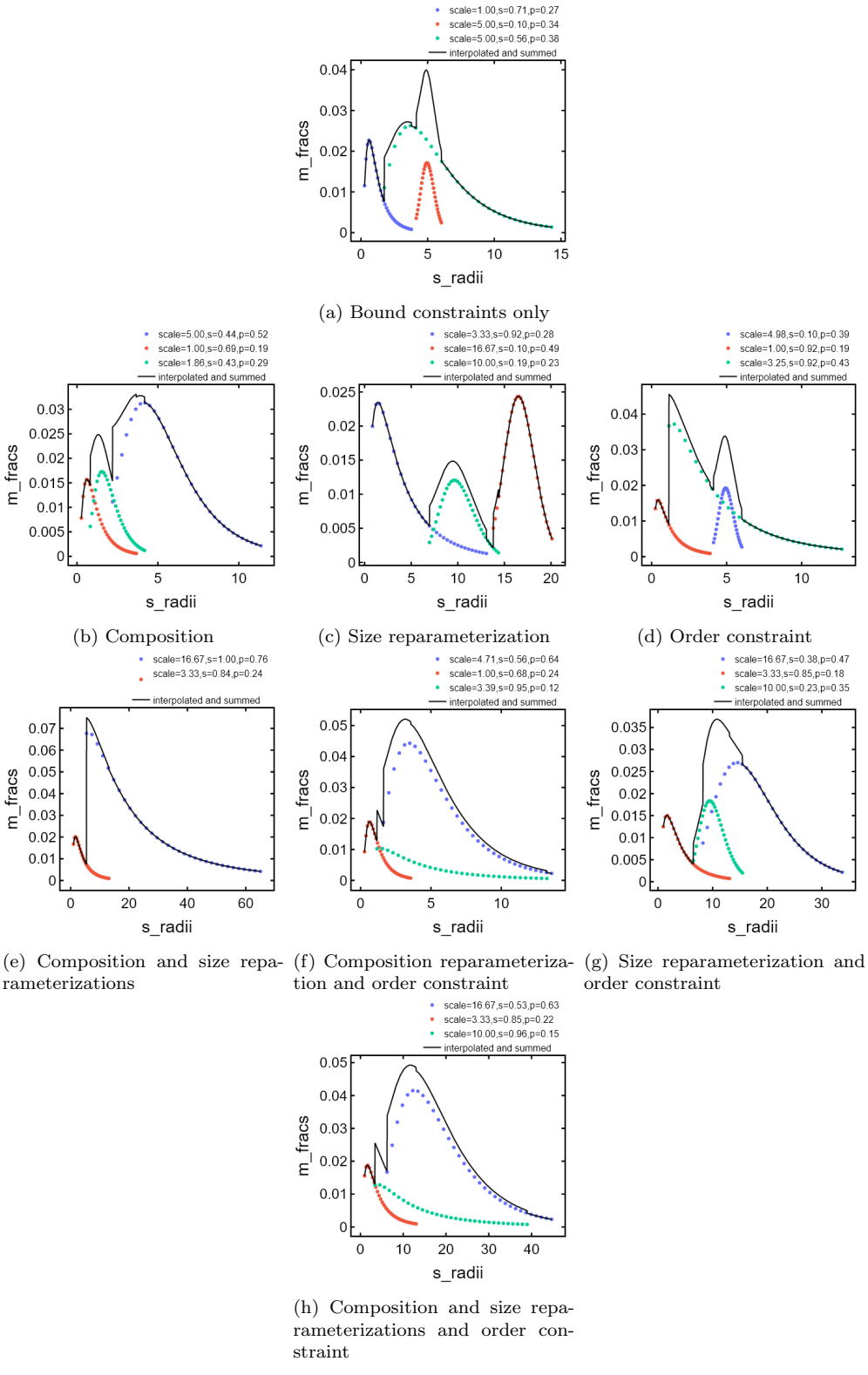
(f) Composition reparameterization and order constraint

(g) Size reparameterization and order constraint

(h) Composition and size reparameterizations and order constraint

Figure S10: Solutions visualized as summed distributions of each of the three particle distributions for each of the eight search spaces. Seed is 10.

S3.4. Seed=11



(a) Bound constraints only

(b) Composition

(c) Size reparameterization

(d) Order constraint

(e) Composition and size reparameterizations

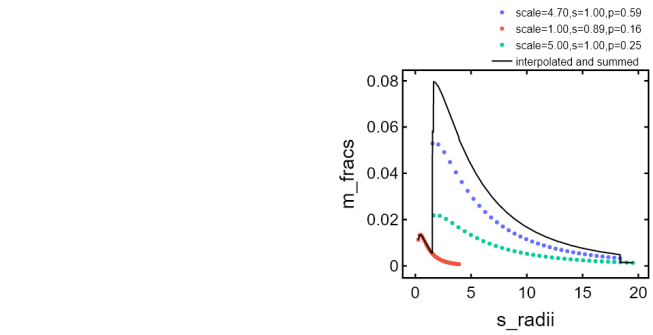
(f) Composition reparameterization and order constraint

(g) Size reparameterization and order constraint

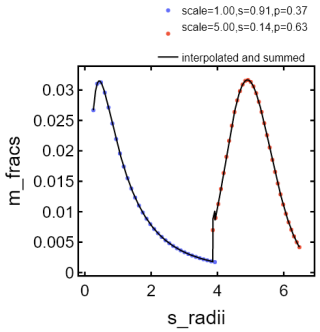
(h) Composition and size reparameterizations and order constraint

Figure S11: Solutions visualized as summed distributions of each of the three particle distributions for each of the eight search spaces. Seed is 11.

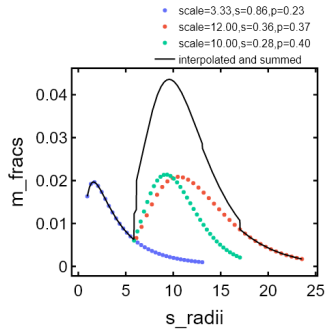
S3.5. Seed=12



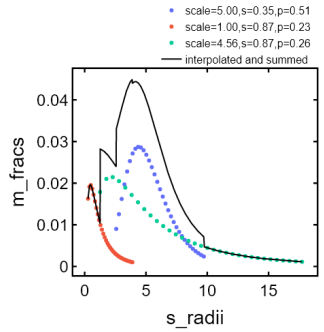
(a) Bound constraints only



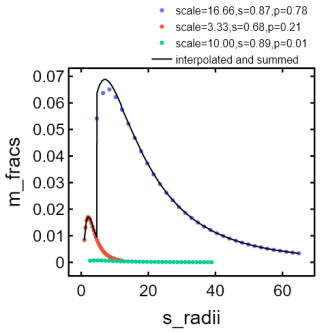
(b) Composition



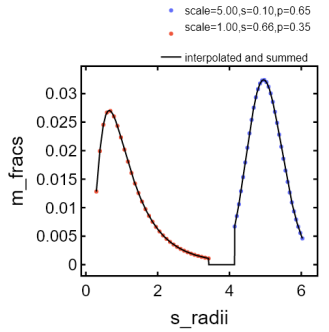
(c) Size reparameterization



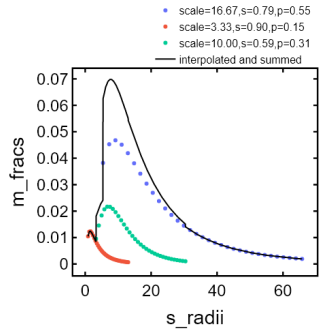
(d) Order constraint



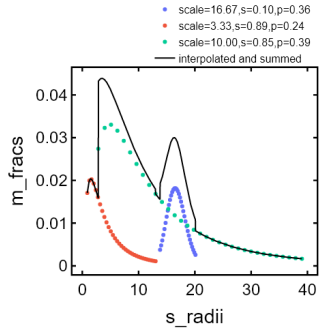
(e) Composition and size reparameterizations



(f) Composition reparameterization and order constraint



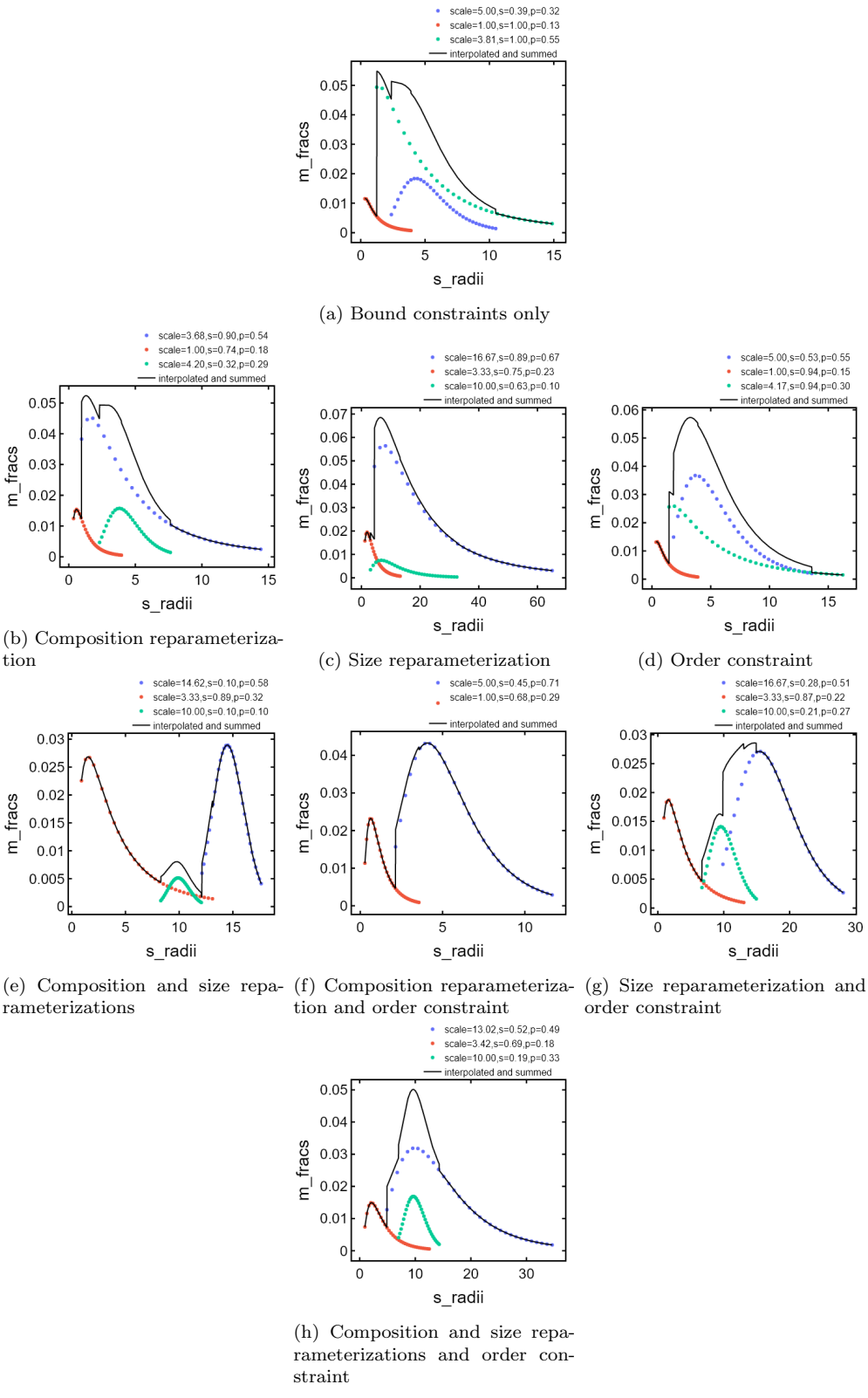
(g) Size reparameterization and order constraint



(h) Composition and size reparameterizations and order constraint

Figure S12: Solutions visualized as summed distributions of each of the three particle distributions for each of the eight search spaces. Seed is 12.

S3.6. Seed=13



(a) Bound constraints only

(b) Composition reparameterization

(c) Size reparameterization

(d) Order constraint

(e) Composition and size reparameterizations

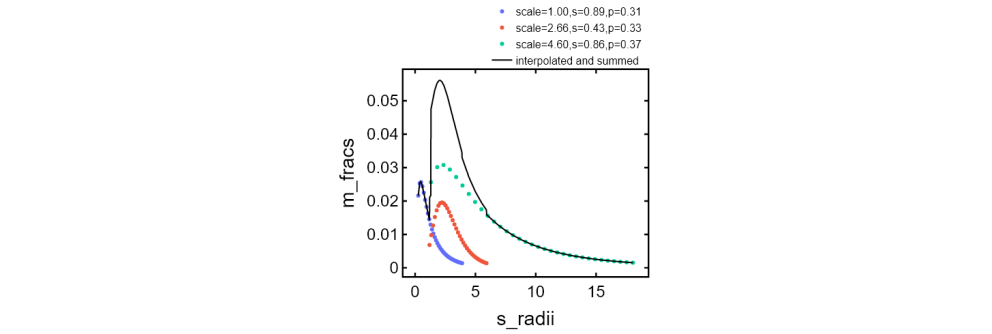
(f) Composition reparameterization and order constraint

(g) Size reparameterization and order constraint

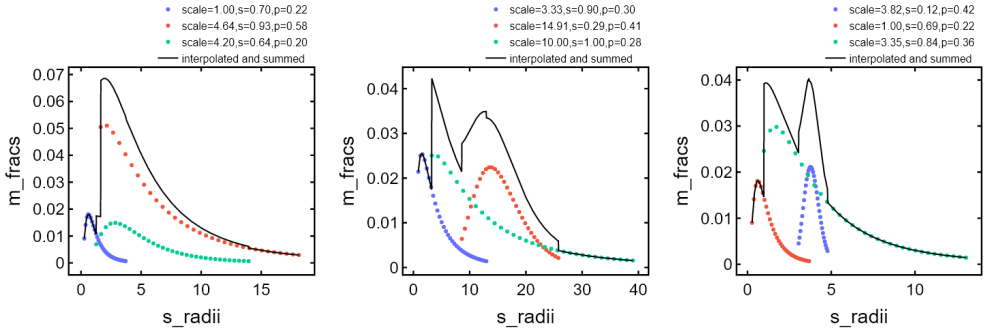
(h) Composition and size reparameterizations and order constraint

Figure S13: Solutions visualized as summed distributions of each of the three particle distributions for each of the eight search spaces. Seed is 13.

S3.7. Seed=14



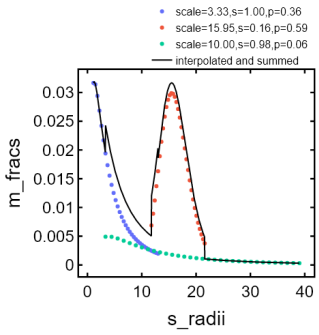
(a) Bound constraints only



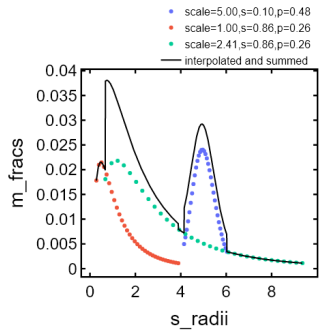
(b) Composition

(c) Size reparameterization

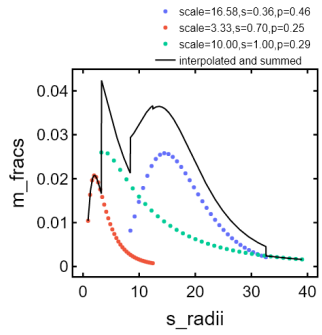
(d) Order constraint



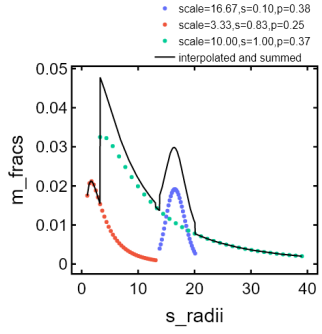
(e) Composition and size reparameterizations



(f) Composition reparameterization and order constraint



(g) Size reparameterization and order constraint



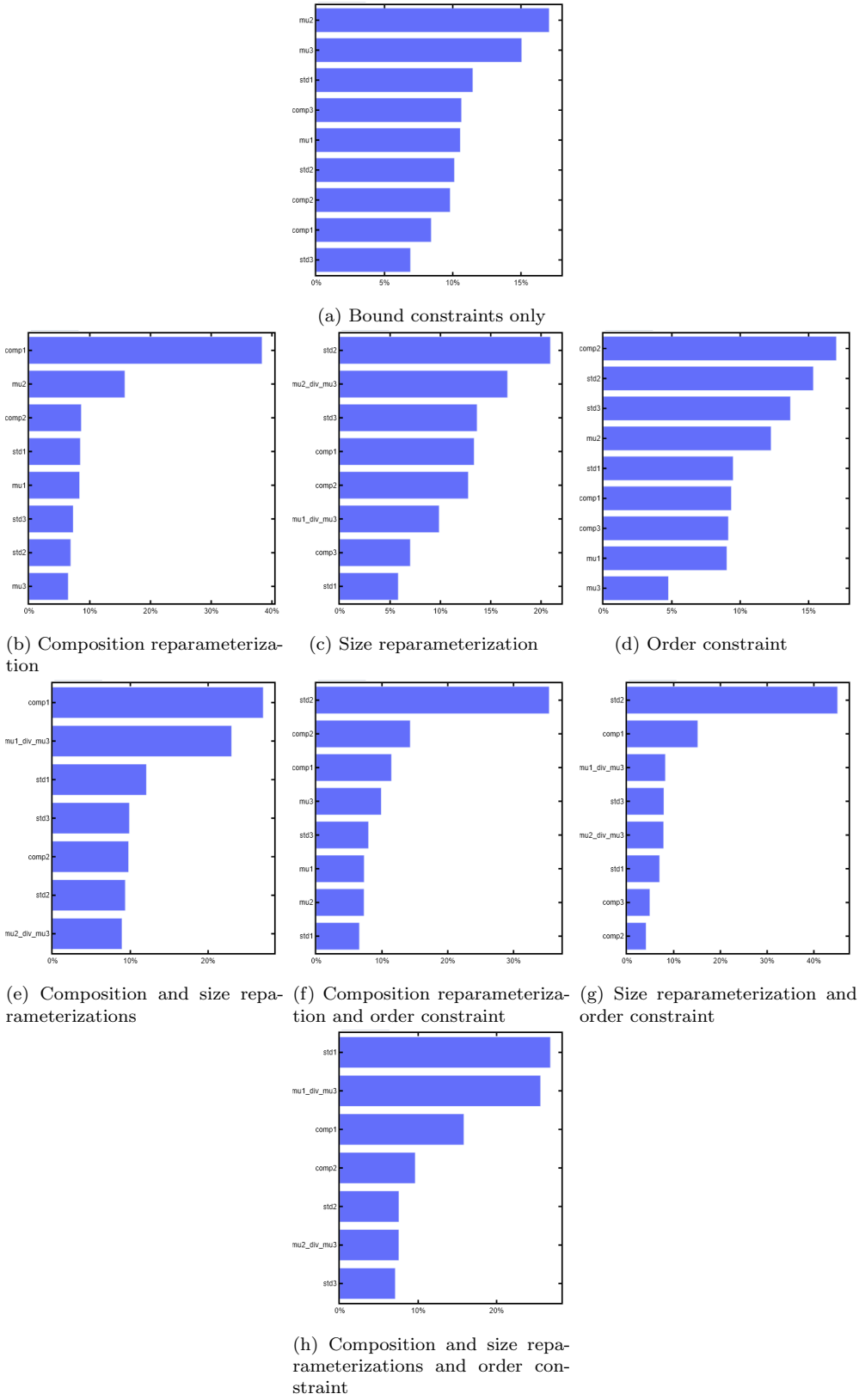
(h) Composition and size reparameterizations and order constraint

Figure S14: Solutions visualized as summed distributions of each of the three particle distributions for each of the eight search spaces. Seed is 14.

S4. Feature Importances

Feature importances for each of the repeat, seeded runs are given in Figures S15–S19.

S4.1. Seed=10



(a) Bound constraints only

(b) Composition reparameterization

(c) Size reparameterization

(d) Order constraint

(e) Composition and size reparameterizations

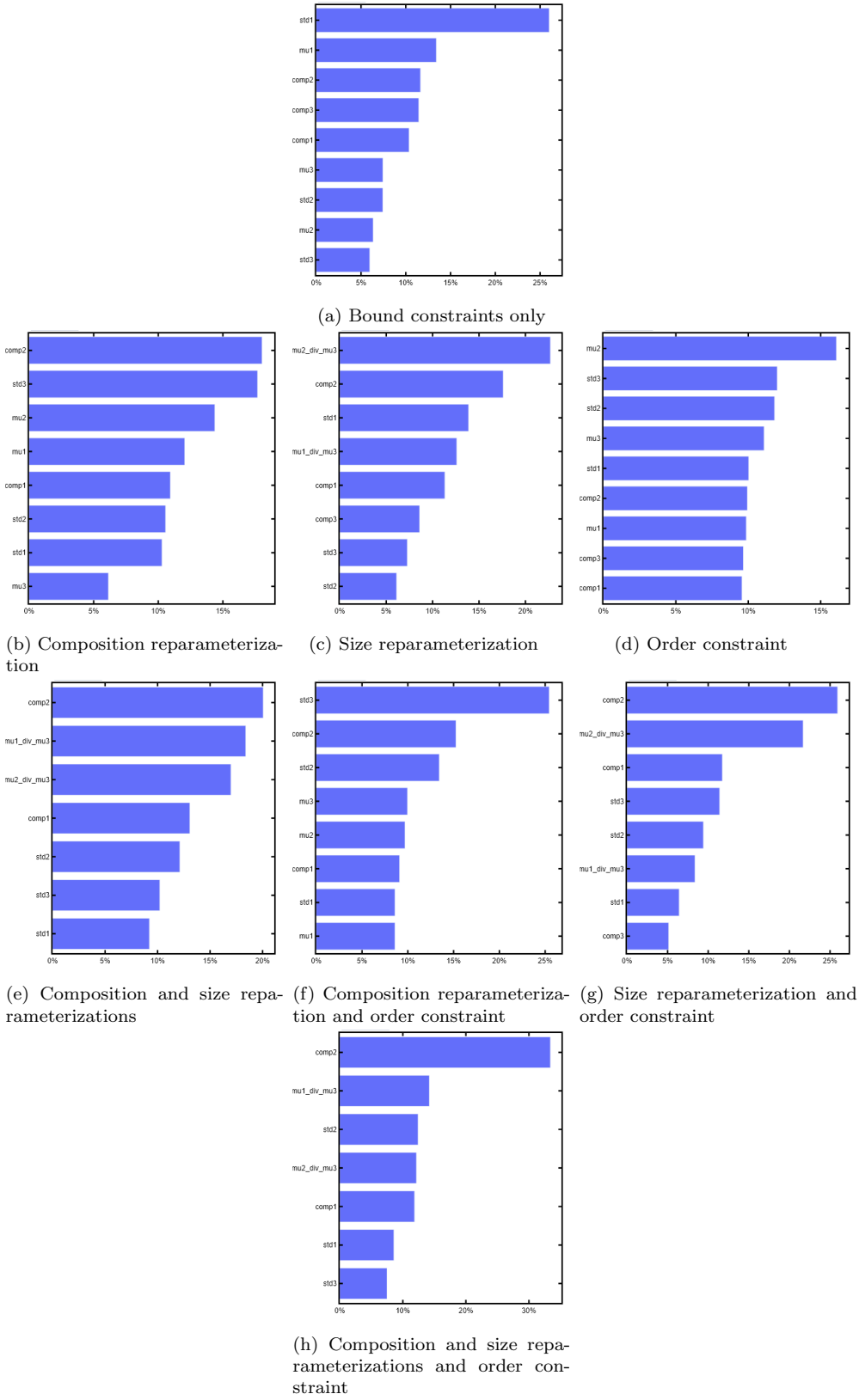
(f) Composition reparameterization and order constraint

(g) Size reparameterization and order constraint

(h) Composition and size reparameterizations and order constraint

Figure S15: Feature importances for the eight search spaces using a seed of 10.

S4.2. Seed=11



(a) Bound constraints only

(b) Composition reparameterization

(c) Size reparameterization

(d) Order constraint

(e) Composition and size reparameterizations

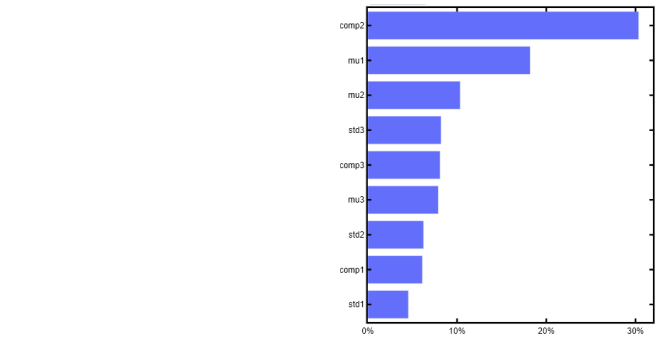
(f) Composition reparameterization and order constraint

(g) Size reparameterization and order constraint

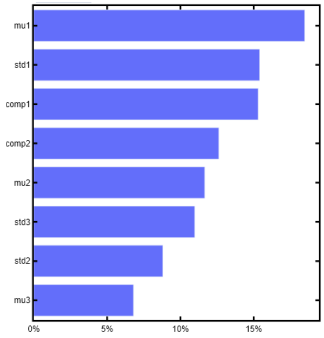
(h) Composition and size reparameterizations and order constraint

Figure S16: Feature importances for the eight search spaces using a seed of 11.

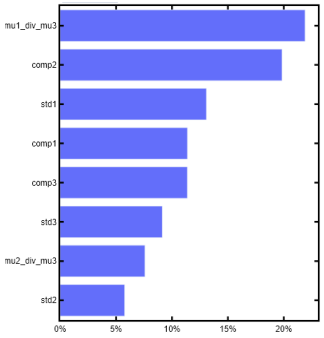
S4.3. Seed=12



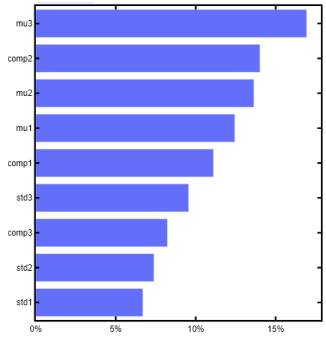
(a) Bound constraints only



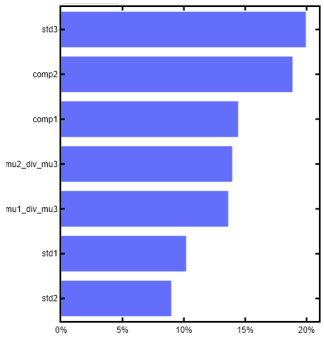
(b) Composition reparameterization



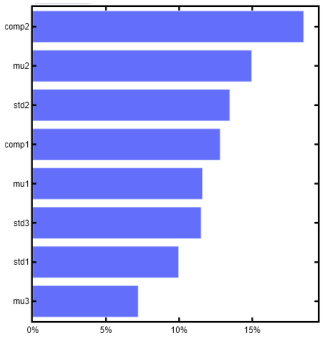
(c) Size reparameterization



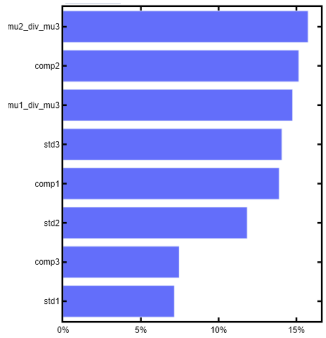
(d) Order constraint



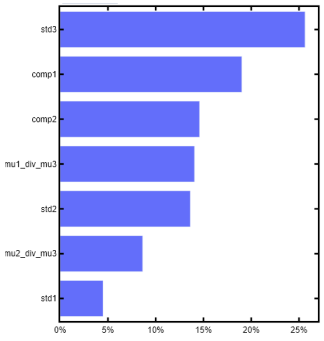
(e) Composition and size reparameterizations



(f) Composition reparameterization and order constraint



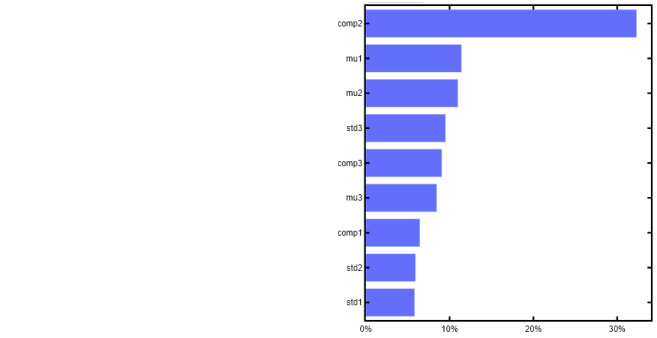
(g) Size reparameterization and order constraint



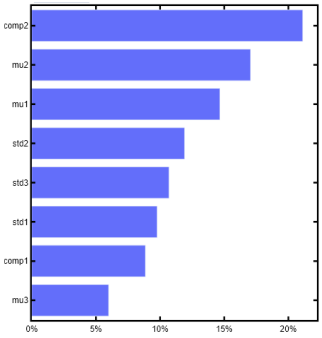
(h) Composition and size reparameterizations and order constraint

Figure S17: Feature importances for the eight search spaces using a seed of 12.

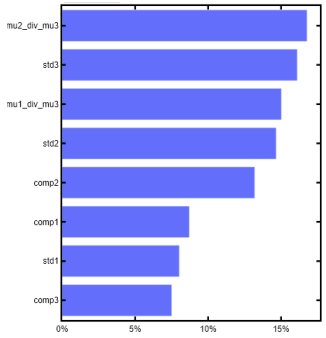
S4.4. Seed=13



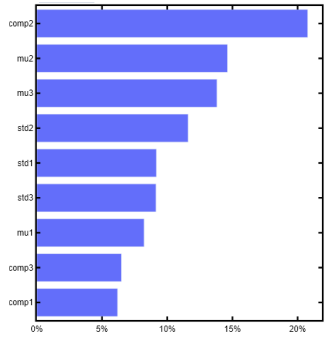
(a) Bound constraints only



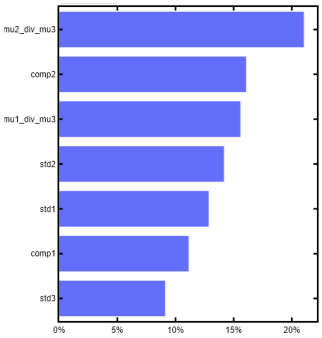
(b) Composition reparameterization



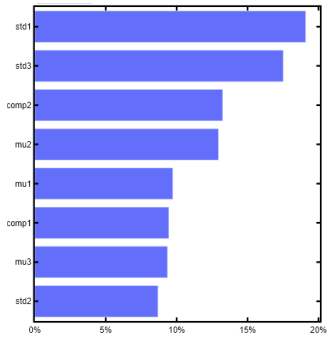
(c) Size reparameterization



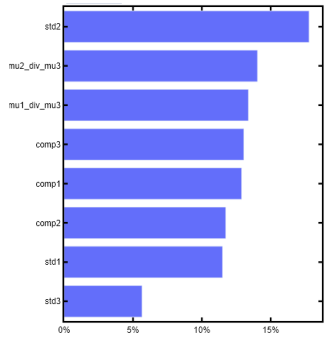
(d) Order constraint



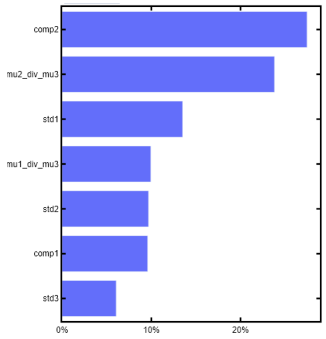
(e) Composition and size reparameterizations



(f) Composition reparameterization and order constraint



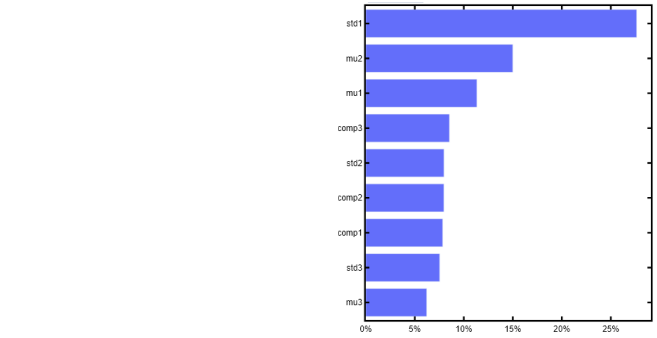
(g) Size reparameterization and order constraint



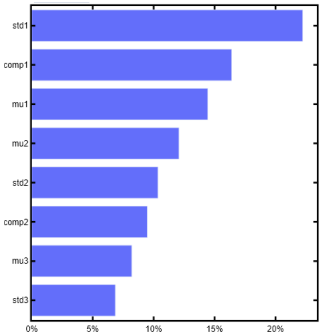
(h) Composition and size reparameterizations and order constraint

Figure S18: Feature importances for the eight search spaces using a seed of 13.

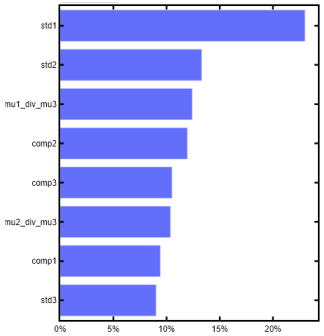
S4.5. Seed=14



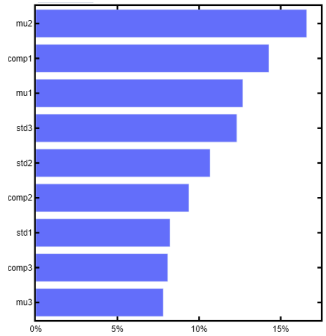
(a) Bound constraints only



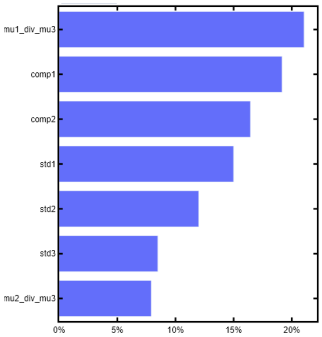
(b) Composition reparameterization



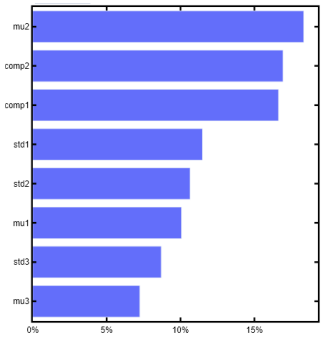
(c) Size reparameterization



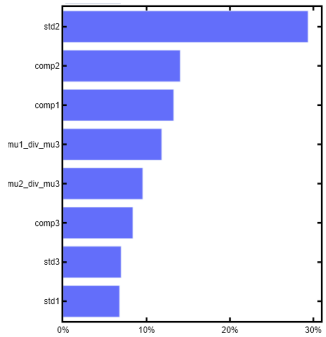
(d) Order constraint



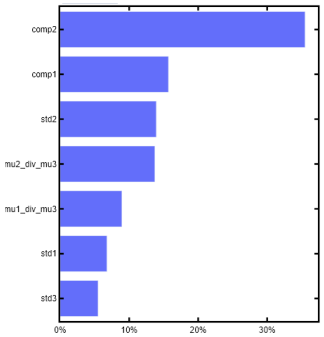
(e) Composition and size reparameterizations



(f) Composition reparameterization and order constraint



(g) Size reparameterization and order constraint



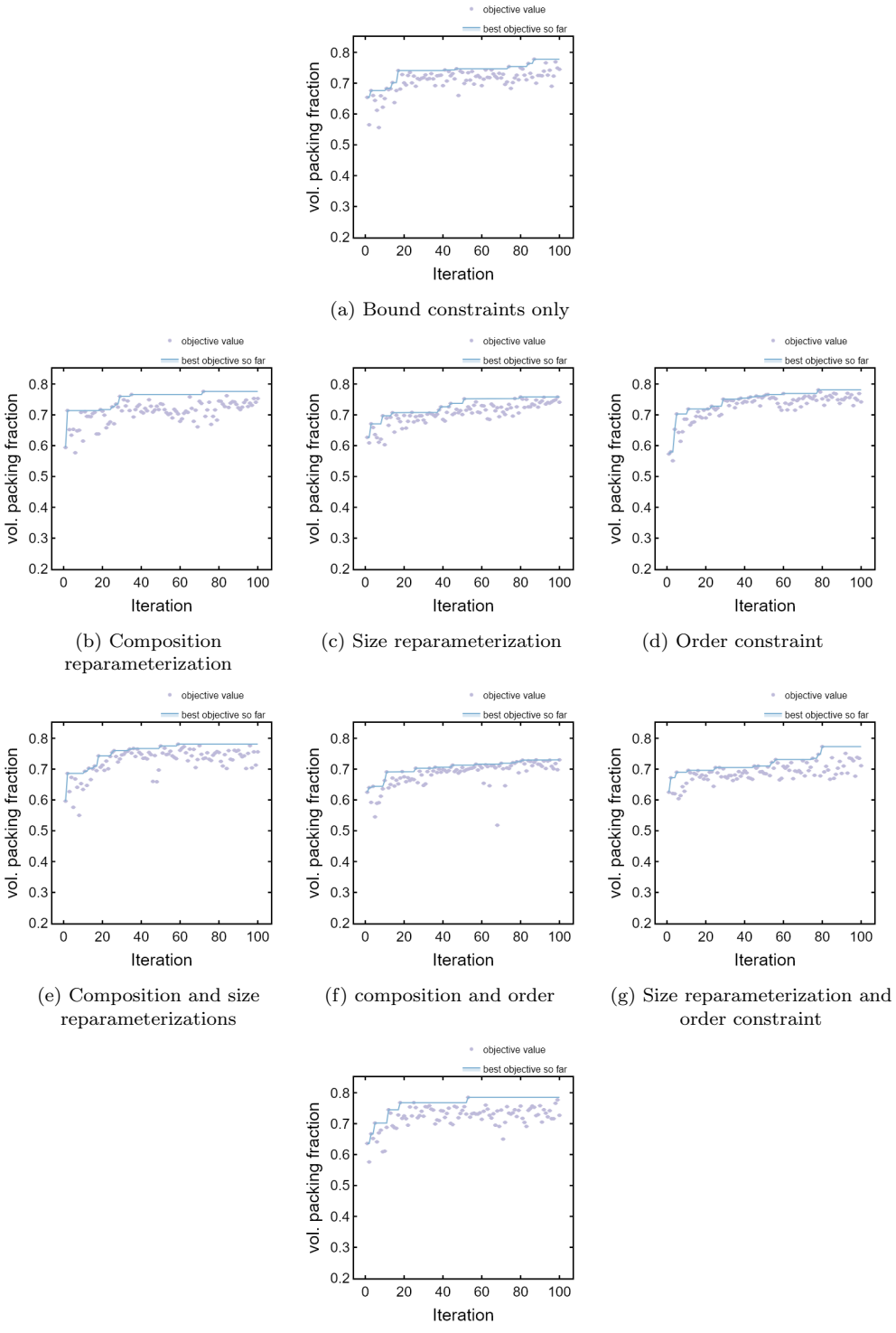
(h) Composition and size reparameterizations and order constraint

Figure S19: Feature importances for the eight search spaces using a seed of 14.

S5. Best Objective vs. Iteration

Best objective vs. iteration plots for each of the repeat, seeded runs are given in Figures S20–S24.

S5.1. Seed=10



(a) Bound constraints only

(b) Composition reparameterization

(c) Size reparameterization

(d) Order constraint

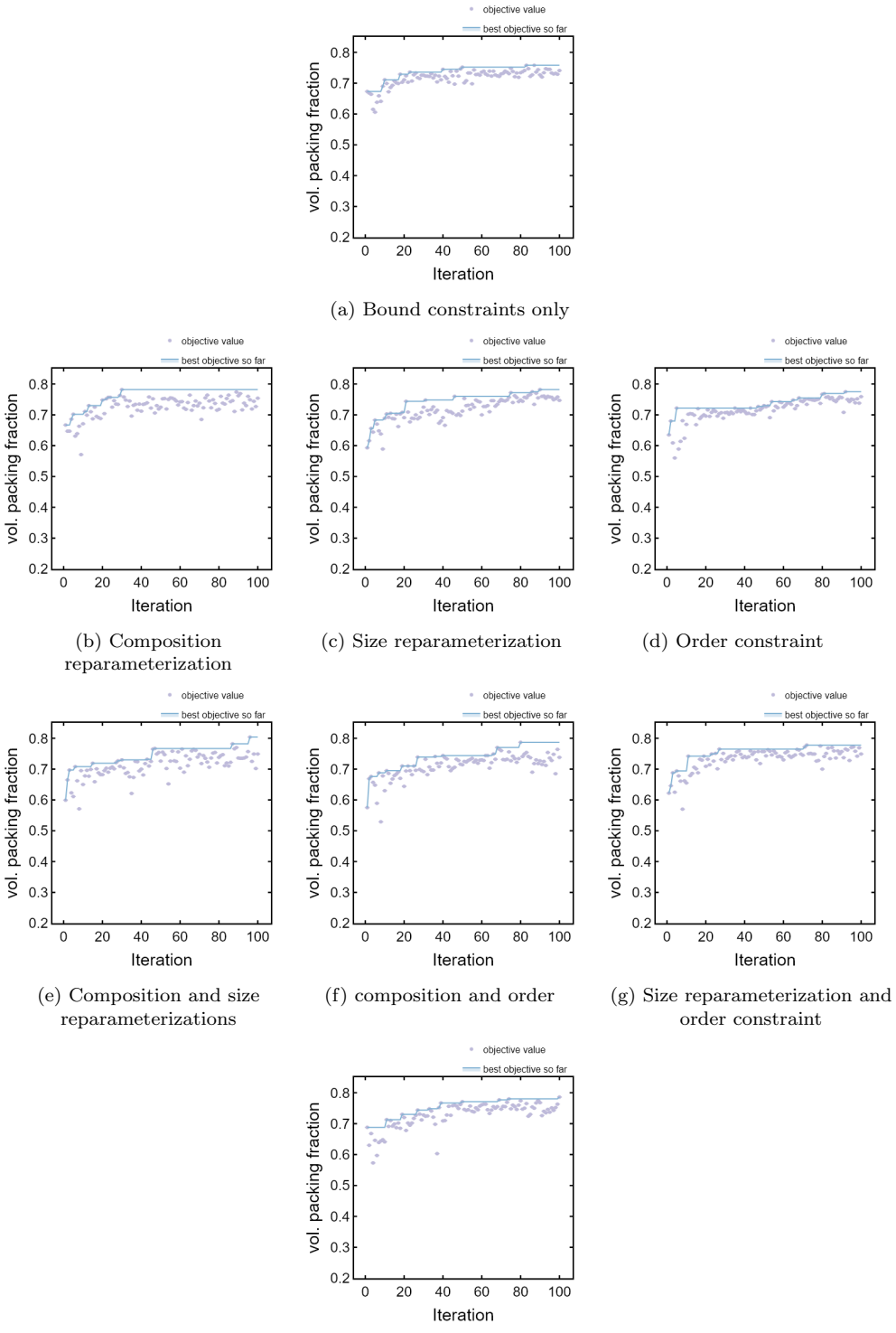
(e) Composition and size reparameterizations

(f) composition and order

(g) Size reparameterization and order constraint

(h) Composition and size reparameterizations and order constraint

Figure S20: best objective vs. iteration for the eight search spaces using a random seed of 10.



(a) Bound constraints only

(b) Composition reparameterization

(c) Size reparameterization

(d) Order constraint

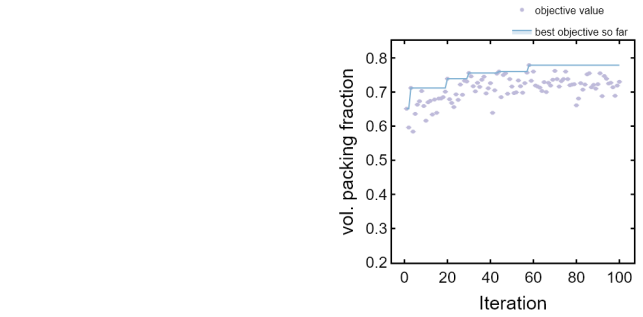
(e) Composition and size reparameterizations

(f) composition and order

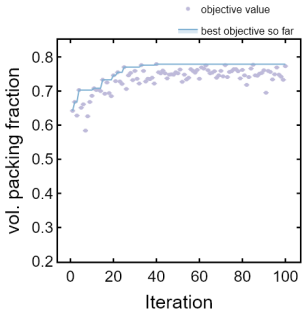
(g) Size reparameterization and order constraint

(h) Composition and size reparameterizations and order constraint

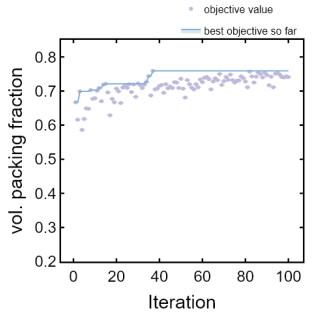
Figure S21: best objective vs. iteration for the eight search spaces using a random seed of 11.



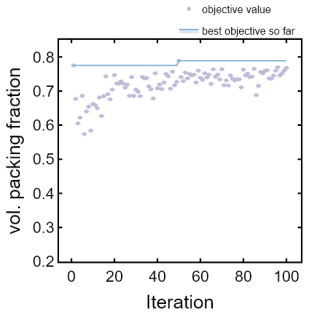
(a) Bound constraints only



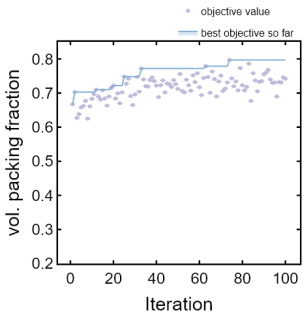
(b) Composition reparameterization



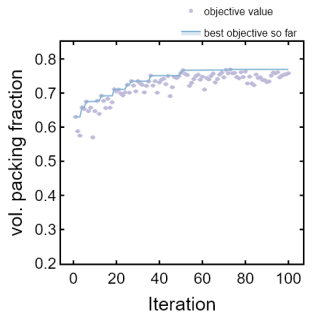
(c) Size reparameterization



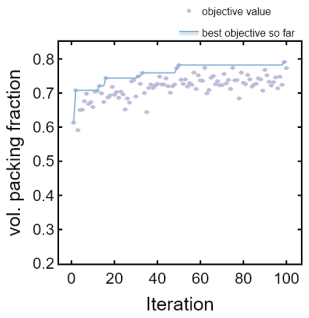
(d) Order constraint



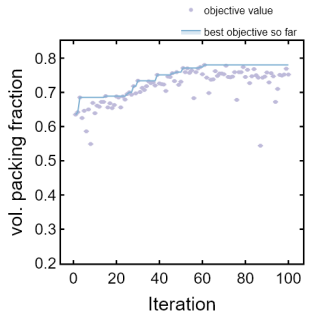
(e) Composition and size reparameterizations



(f) composition and order



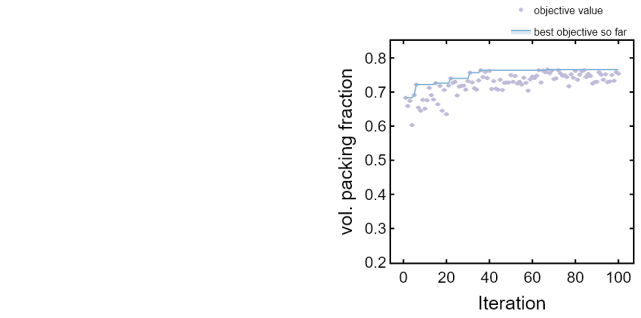
(g) Size reparameterization and order constraint



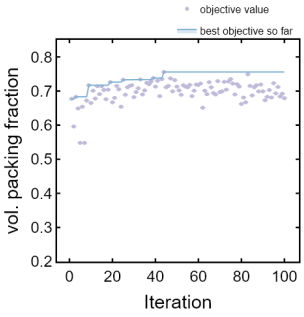
(h) Composition and size reparameterizations and order constraint

Figure S22: best objective vs. iteration for the eight search spaces using a random seed of 12.

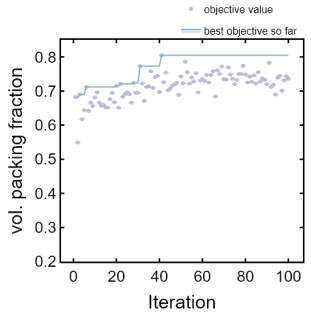
S5.4. Seed=13



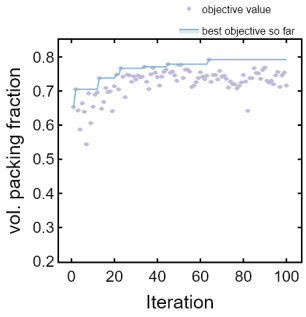
(a) Bound constraints only



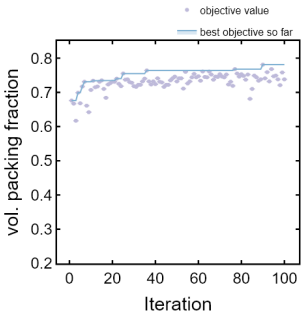
(b) Composition reparameterization



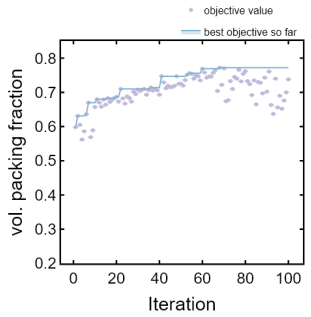
(c) Size reparameterization



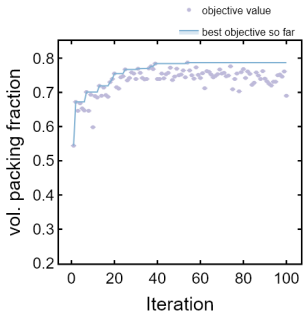
(d) Order constraint



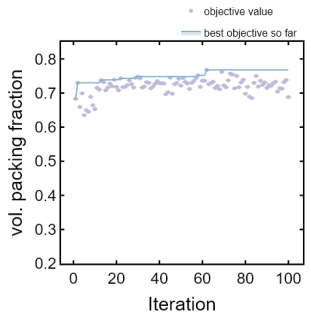
(e) Composition and size reparameterizations



(f) composition and order



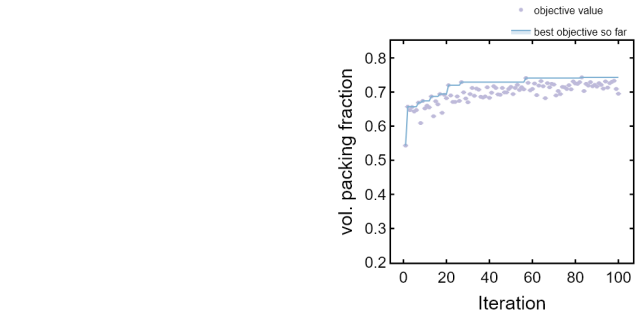
(g) Size reparameterization and order constraint



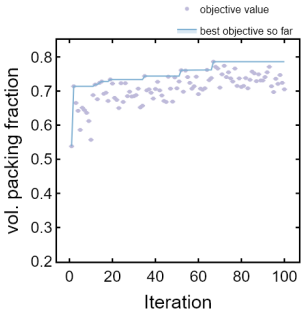
(h) Composition and size reparameterizations and order constraint

Figure S23: best objective vs. iteration for the eight search spaces using a random seed of 13.

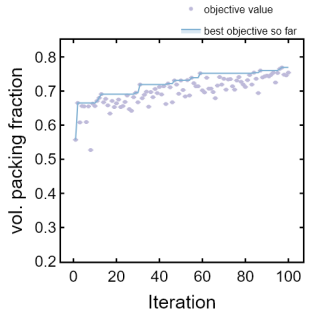
S5.5. Seed=14



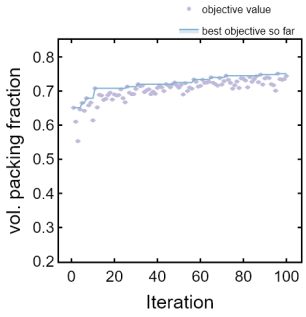
(a) Bound constraints only



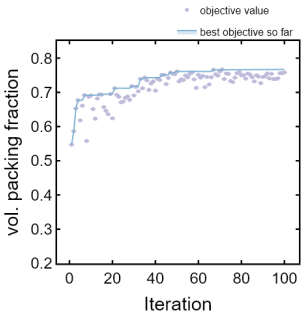
(b) Composition reparameterization



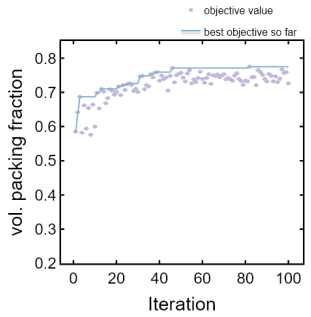
(c) Size reparameterization



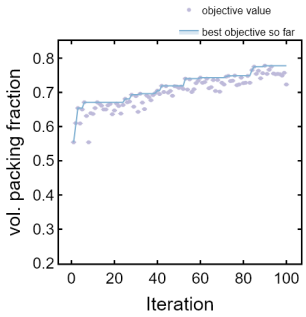
(d) Order constraint



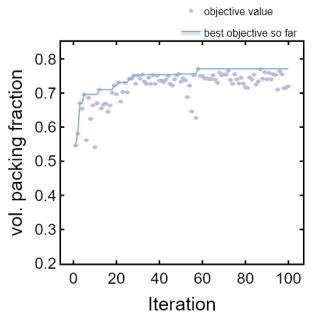
(e) Composition and size reparameterizations



(f) composition and order



(g) Size reparameterization and order constraint



(h) Composition and size reparameterizations and order constraint

Figure S24: best objective vs. iteration for the eight search spaces using a random seed of 14.

S6. Cross Validation Results

Leave-one-out cross-validation (LOO-CV) results for all optimizations are given in Figure S25. LOO-CV results for each of the repeat, seeded runs are given in Figures S26–S30.

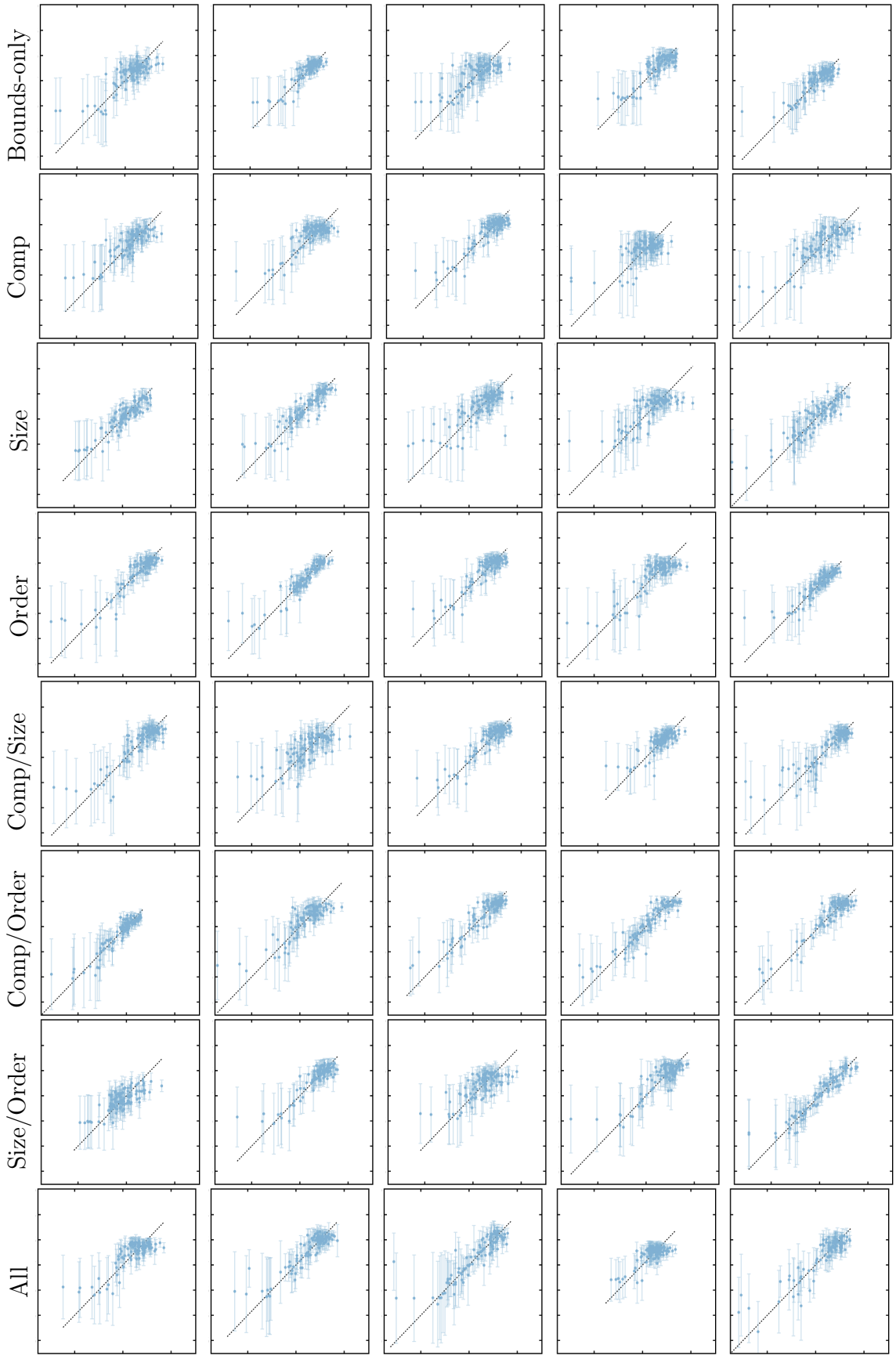
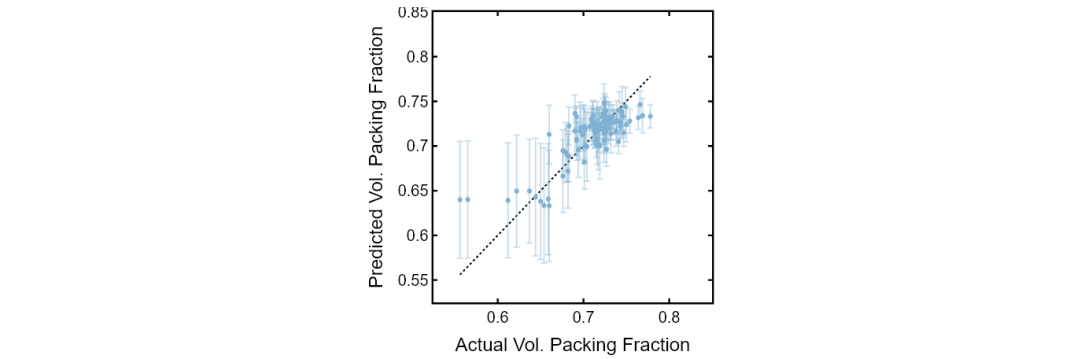
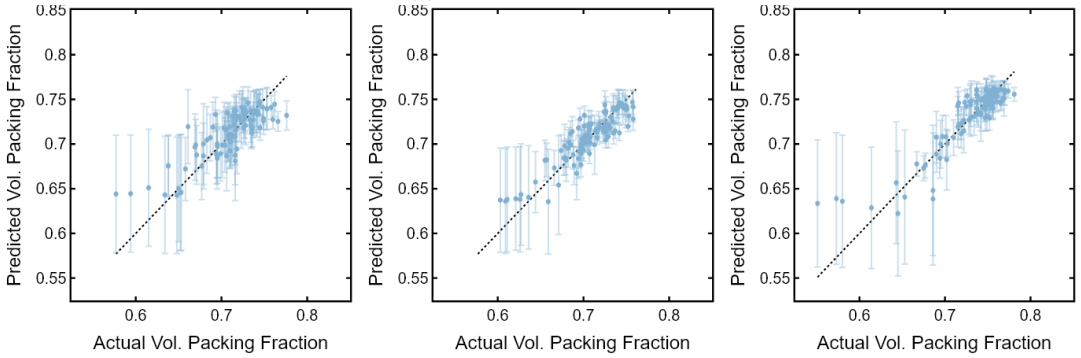


Figure S25: Predicted vs. actual (y vs. x-axes) LOO-CV for the eight search spaces (rows) across five seeded runs (columns).

S6.1. Seed=10



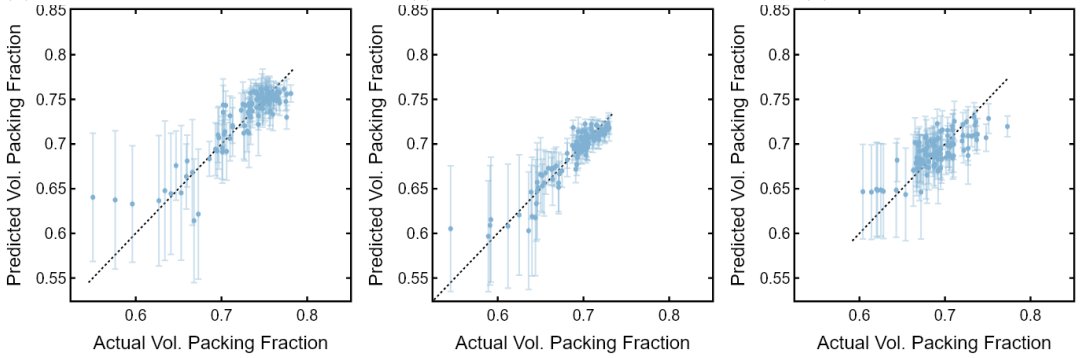
(a) Bound constraints only



(b) Composition reparameterization

(c) Size reparameterization

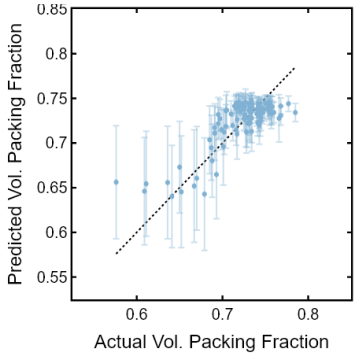
(d) Order constraint



(e) Composition and size reparameterizations

(f) Composition reparameterization and order constraint

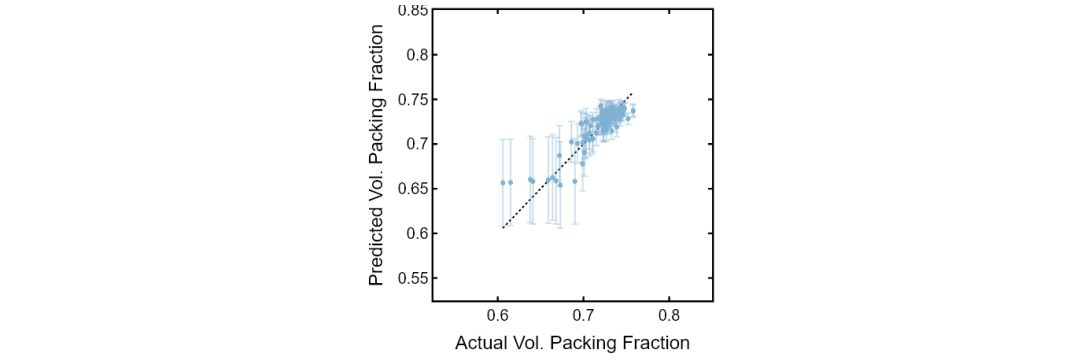
(g) Size reparameterization and order constraint



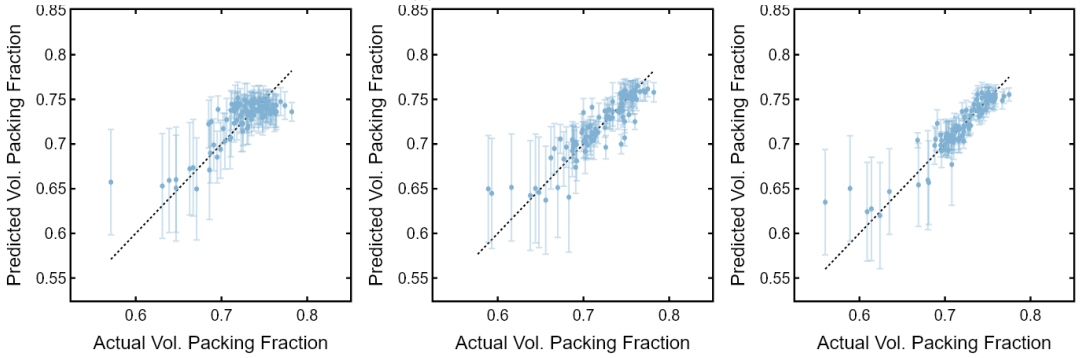
(h) Composition and size reparameterizations and order constraint

Figure S26: LOO-CV results. The first 10 iterations were Sobol points, and the remaining 90 were Bayesian iterations. Seed is 10.

S6.2. Seed=11



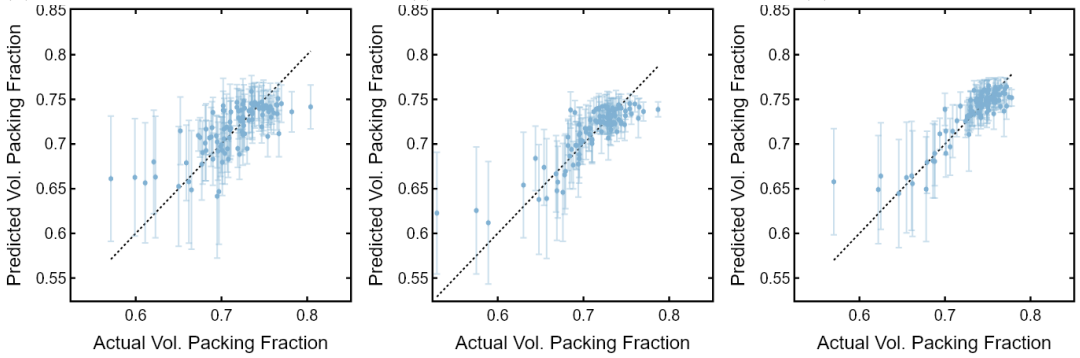
(a) Bound constraints only



(b) Composition reparameterization

(c) Size reparameterization

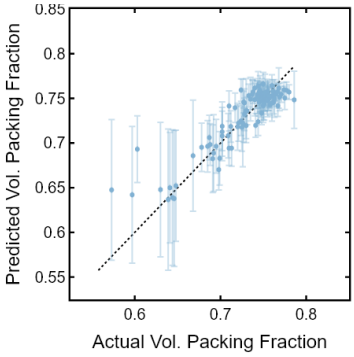
(d) Order constraint



(e) Composition and size reparameterizations

(f) Composition reparameterization and order constraint

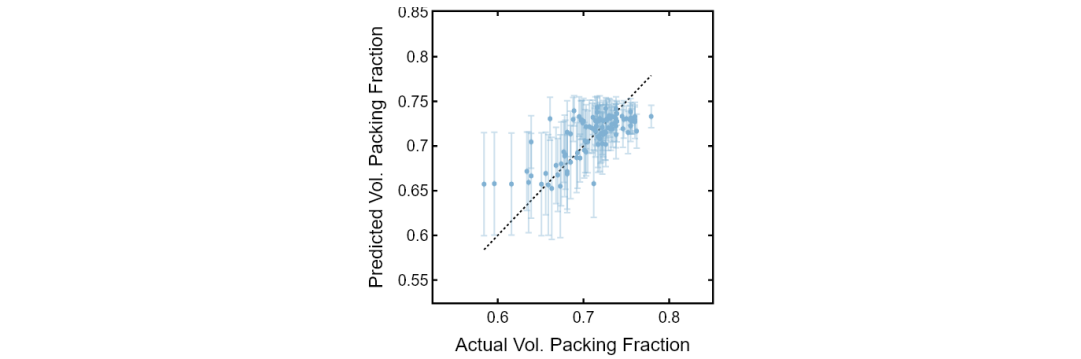
(g) Size reparameterization and order constraint



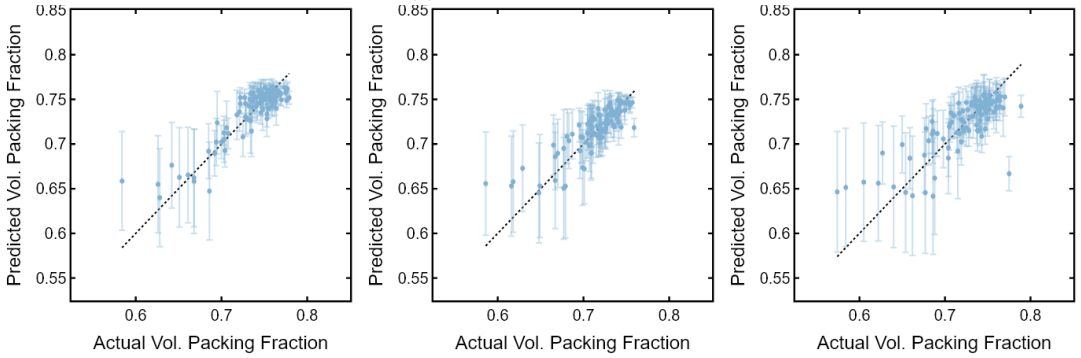
(h) Composition and size reparameterizations and order constraint

Figure S27: LOO-CV results. The first 10 iterations were Sobol points, and the remaining 90 were Bayesian iterations. Seed is 11.

S6.3. Seed=12



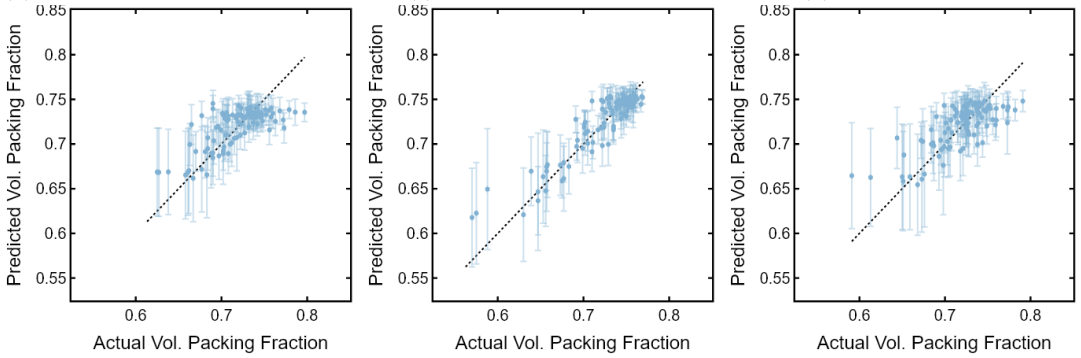
(a) Bound constraints only



(b) Composition reparameterization

(c) Size reparameterization

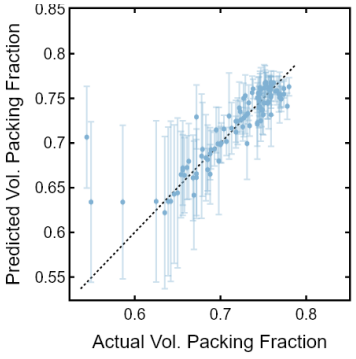
(d) Order constraint



(e) Composition and size reparameterizations

(f) Composition reparameterization and order constraint

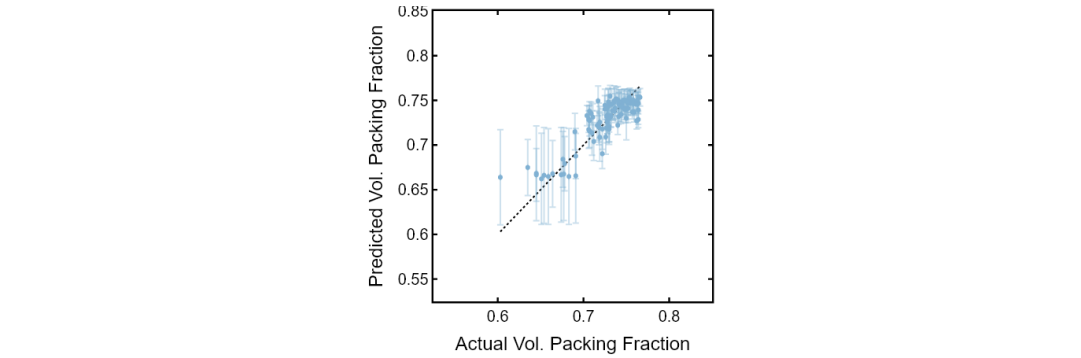
(g) Size reparameterization and order constraint



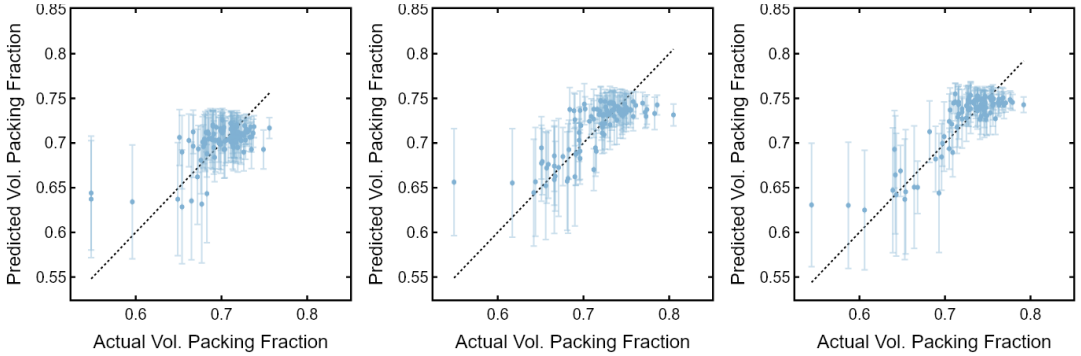
(h) Composition and size reparameterizations and order constraint

Figure S28: LOO-CV results. The first 10 iterations were Sobol points, and the remaining 90 were Bayesian iterations. Seed is 12.

S6.4. Seed=13



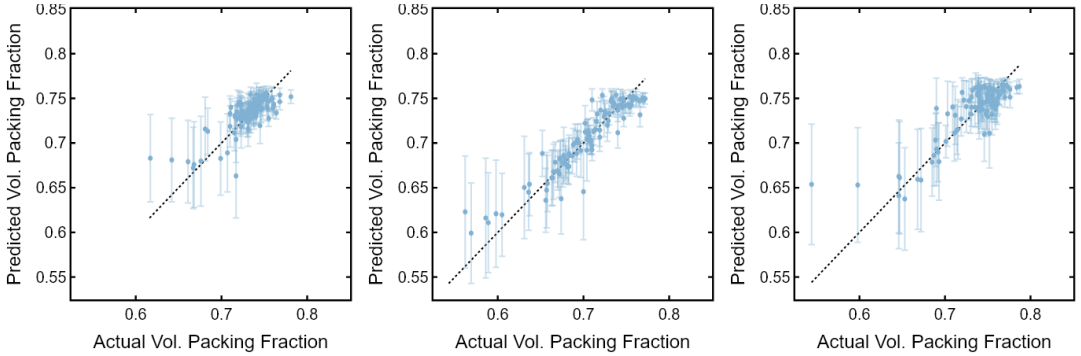
(a) Bound constraints only



(b) Composition reparameterization

(c) Size reparameterization

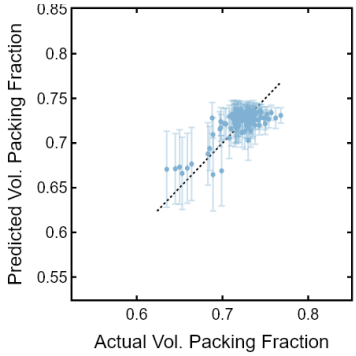
(d) Order constraint



(e) Composition and size reparameterizations

(f) Composition reparameterization and order constraint

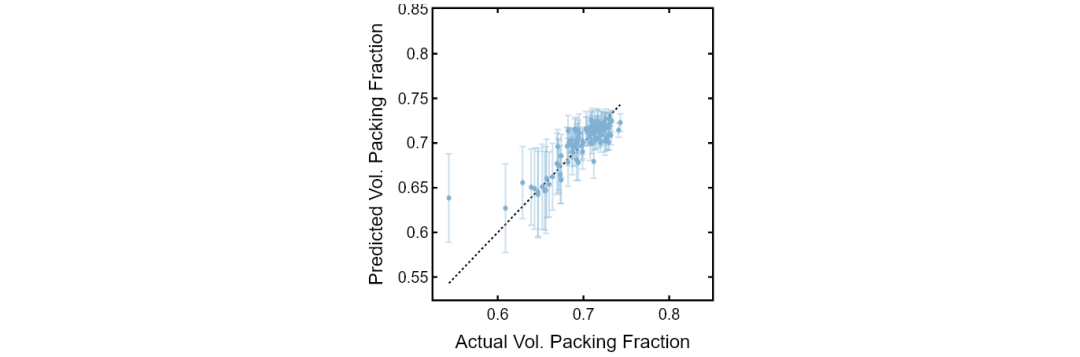
(g) Size reparameterization and order constraint



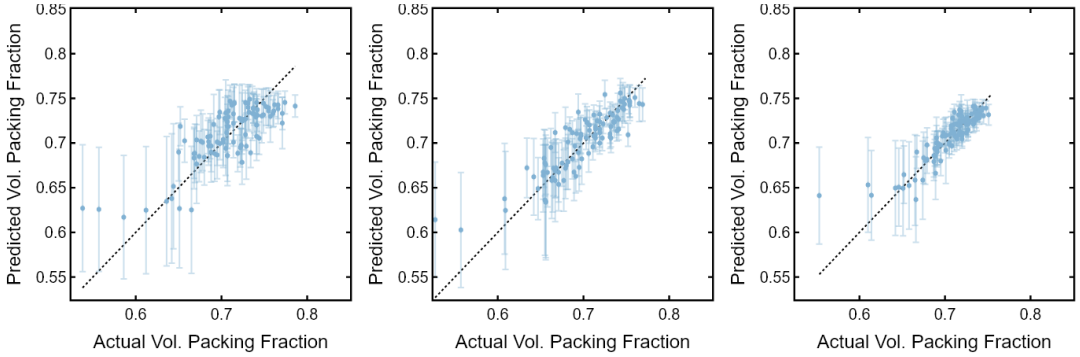
(h) Composition and size reparameterizations and order constraint

Figure S29: LOO-CV results. The first 10 iterations were Sobol points, and the remaining 90 were Bayesian iterations. Seed is 13.

S6.5. Seed=14



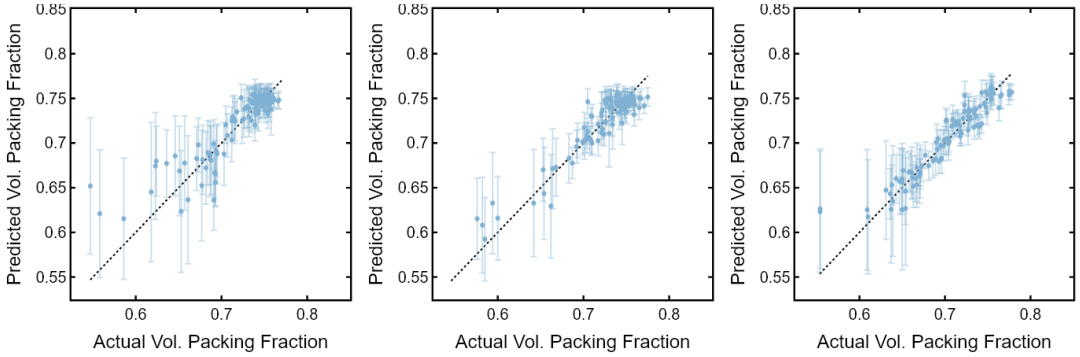
(a) Bound constraints only



(b) Composition reparameterization

(c) Size reparameterization

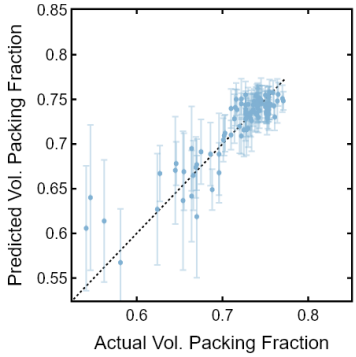
(d) Order constraint



(e) Composition and size reparameterizations

(f) Composition reparameterization and order constraint

(g) Size reparameterization and order constraint



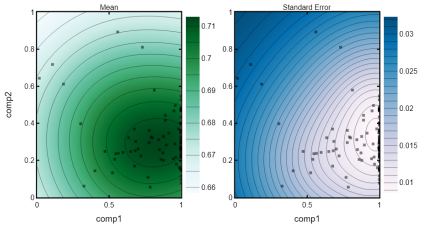
(h) Composition and size reparameterizations and order constraint

Figure S30: LOO-CV results. The first 10 iterations were Sobol points, and the remaining 90 were Bayesian iterations. Seed is 14.

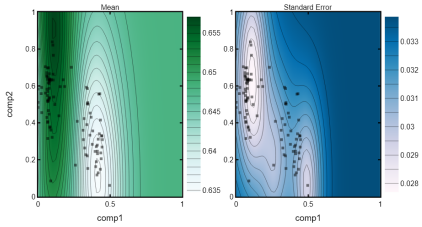
S7. 2D Contours through Model Parameter Space (Comp1 and Comp2)

2D contour plots for the first two compositional variables are given for each of the repeat, seeded runs (except for seed=10, which is present in the main paper - see Figure S31) in Figures S32–S35.

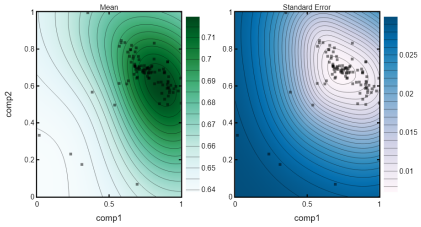
S7.1. Seed=10



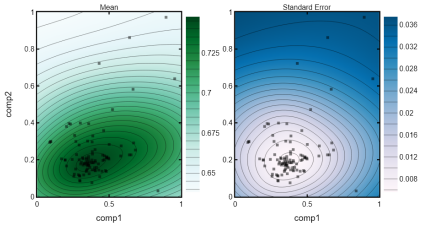
(a) Bound constraints only



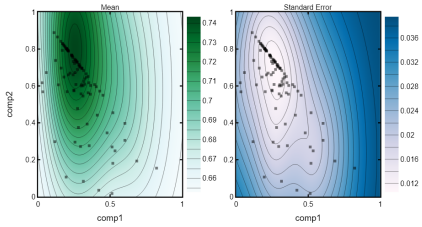
(b) Composition reparameterization



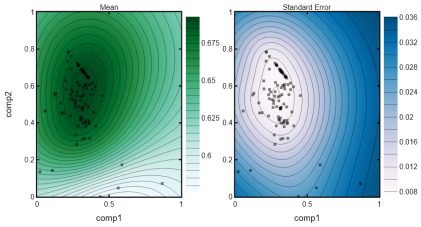
(c) Size reparameterization



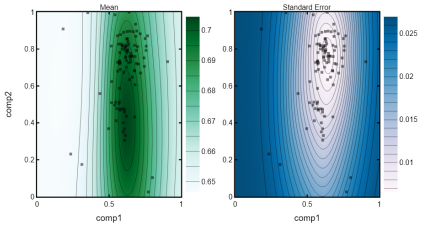
(d) Order constraint



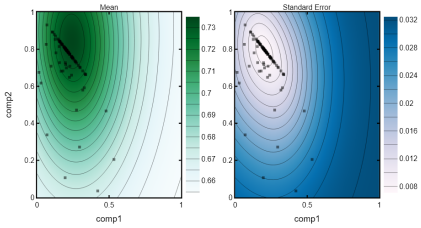
(e) Composition and size reparameterizations



(f) Composition reparameterization and order constraint

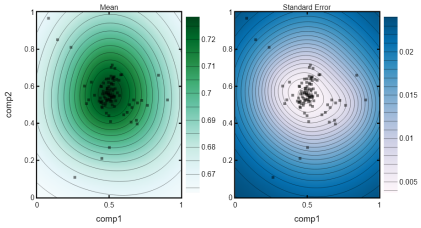


(g) Size reparameterization and order constraint

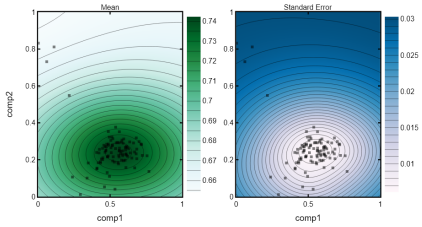


(h) Composition and size reparameterizations and order constraint

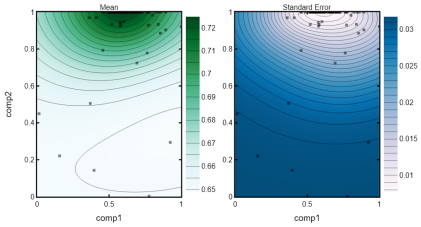
Figure S31: Two-dimensional (2D) contour plots of the first two fractional prevalences (compositions) for each of the 8 constraint combinations. Because the size constraint affects the search space bounds and because the compositional constraint introduces a bias in the sampling, each of these will have fixed the other parameters at somewhat different values. Seed is 10.



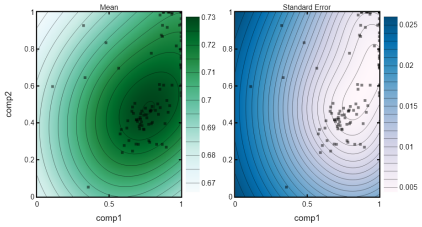
(a) Bound constraints only



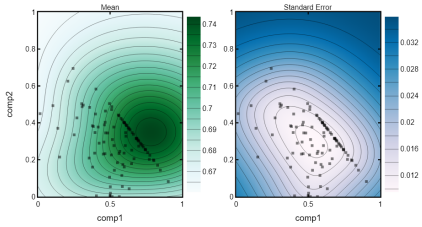
(b) Composition reparameterization



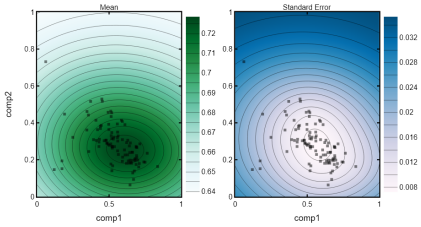
(c) Size reparameterization



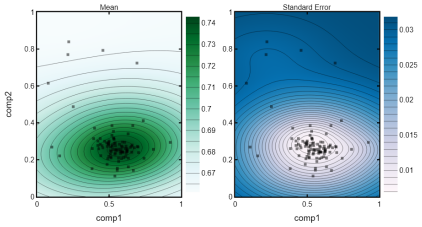
(d) Order constraint



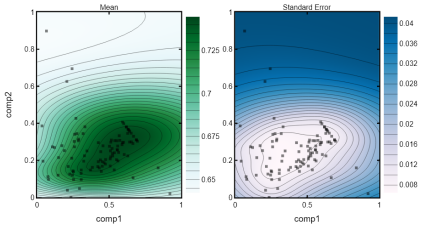
(e) Composition and size reparameterizations



(f) Composition reparameterization and order constraint

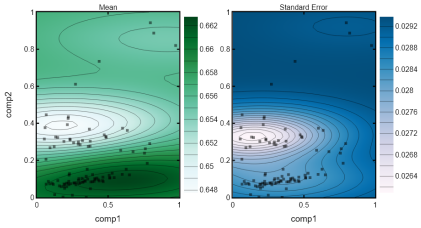


(g) Size reparameterization and order constraint

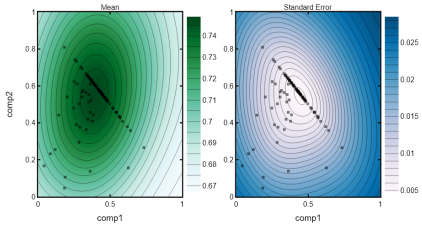


(h) Composition and size reparameterizations and order constraint

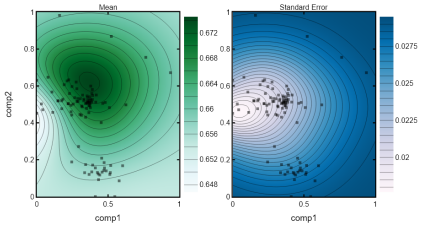
Figure S32: Two-dimensional (2D) contour plots of the first two fractional prevalences (compositions) for each of the 8 constraint combinations. Because the size constraint affects the search space bounds and because the compositional constraint introduces a bias in the sampling, each of these will have fixed the other parameters at somewhat different values. Seed is 11.



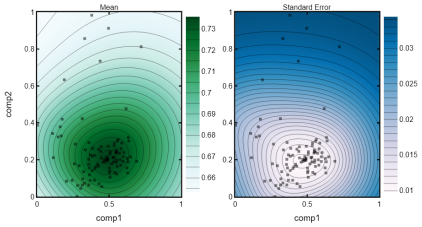
(a) Bound constraints only



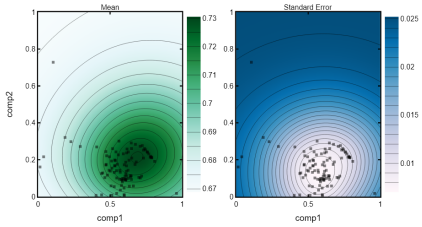
(b) Composition reparameterization



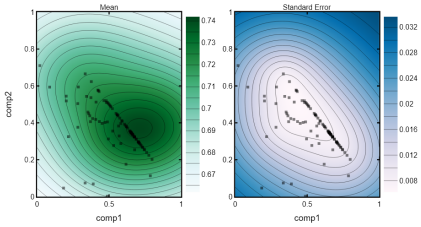
(c) Size reparameterization



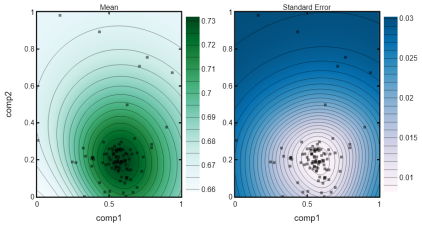
(d) Order constraint



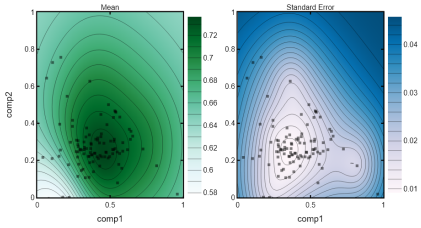
(e) Composition and size reparameterizations



(f) Composition reparameterization and order constraint

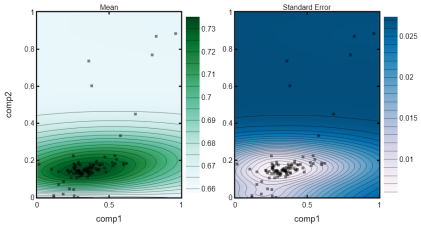


(g) Size reparameterization and order constraint

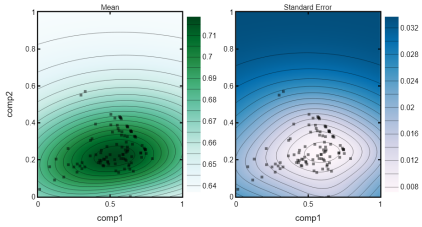


(h) Composition and size reparameterizations and order constraint

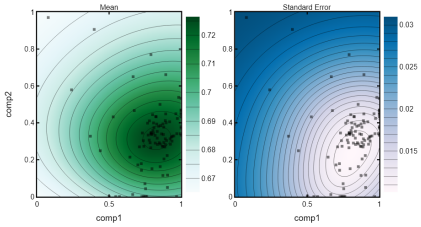
Figure S33: Two-dimensional (2D) contour plots of the first two fractional prevalences (compositions) for each of the 8 constraint combinations. Because the size constraint affects the search space bounds and because the compositional constraint introduces a bias in the sampling, each of these will have fixed the other parameters at somewhat different values. Seed is 12.



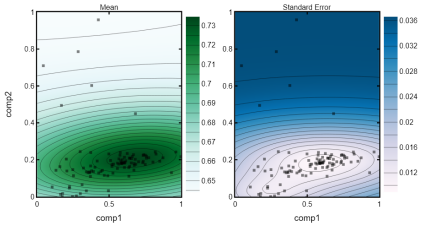
(a) Bound constraints only



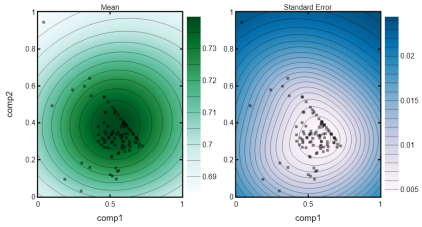
(b) Composition reparameterization



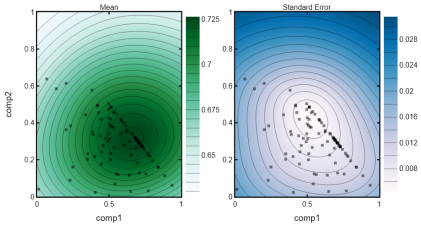
(c) Size reparameterization



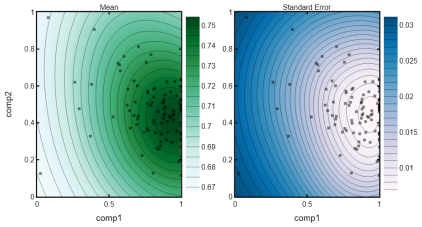
(d) Order constraint



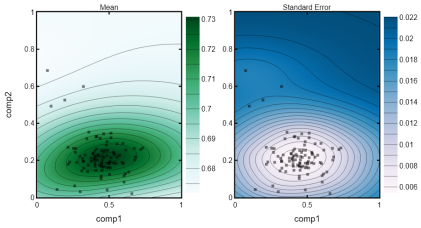
(e) Composition and size reparameterizations



(f) Composition reparameterization and order constraint



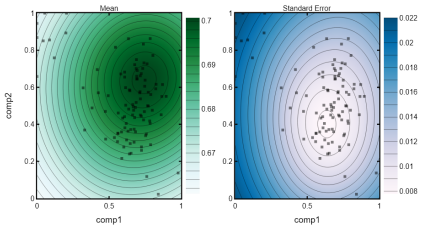
(g) Size reparameterization and order constraint



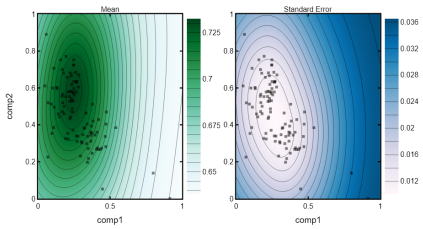
(h) Composition and size reparameterizations and order constraint

Figure S34: Two-dimensional (2D) contour plots of the first two fractional prevalences (compositions) for each of the 8 constraint combinations. Because the size constraint affects the search space bounds and because the compositional constraint introduces a bias in the sampling, each of these will have fixed the other parameters at somewhat different values. Seed is 13.

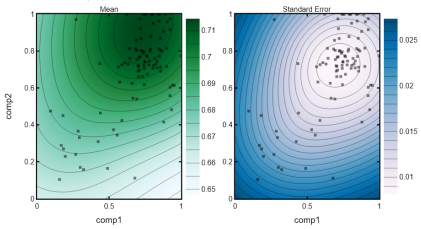
S7.5. Seed=14



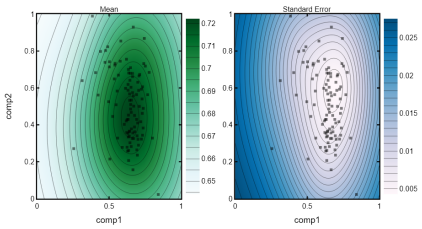
(a) Bound constraints only



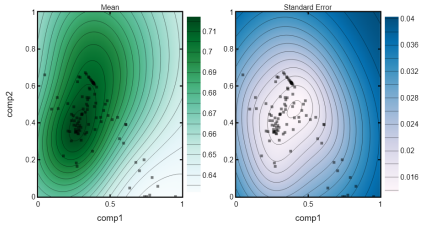
(b) Composition reparameterization



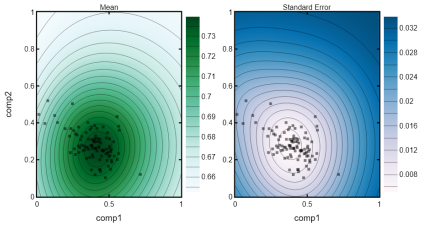
(c) Size reparameterization



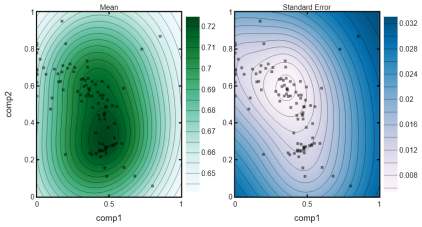
(d) Order constraint



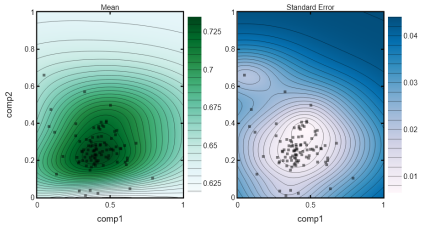
(e) Composition and size reparameterizations



(f) Composition reparameterization and order constraint



(g) Size reparameterization and order constraint



(h) Composition and size reparameterizations and order constraint

Figure S35: Two-dimensional (2D) contour plots of the first two fractional prevalences (compositions) for each of the 8 constraint combinations. Because the size constraint affects the search space bounds and because the compositional constraint introduces a bias in the sampling, each of these will have fixed the other parameters at somewhat different values. Seed is 14.

S8. Ax SearchSpace Objects

Ax `SearchSpace` objects for each of the eight search spaces are provided. These are preceded by a dictionary that define which of the eight search spaces is being used.

S8.1. Composition, Size, and Order

```
{
    "remove_scaling_degeneracy": True,
    "remove_composition_degeneracy": True,
    "use_order_constraint": True,
}

SearchSpace(
    parameters=[
        RangeParameter(name="mu1_div_mu3", parameter_type=FLOAT, range=[0.
                                                    3333333333333333, 1.
                                                    6666666666666667]),
        RangeParameter(name="mu2_div_mu3", parameter_type=FLOAT, range=[0.
                                                    3333333333333333, 1.
                                                    6666666666666667]),
        RangeParameter(name="std1", parameter_type=FLOAT, range=[0.1, 1.0]),
        RangeParameter(name="std2", parameter_type=FLOAT, range=[0.1, 1.0]),
        RangeParameter(name="std3", parameter_type=FLOAT, range=[0.1, 1.0]),
        RangeParameter(name="comp1", parameter_type=FLOAT, range=[0.0, 1.0]),
        RangeParameter(name="comp2", parameter_type=FLOAT, range=[0.0, 1.0]),
    ],
    parameter_constraints=[
        ParameterConstraint(1.0 * comp1 + 1.0 * comp2 <= 1.0),
        OrderConstraint(std1 <= std2),
    ],
)
```

S8.2. Composition and Size

```
{
    "remove_scaling_degeneracy": True,
    "remove_composition_degeneracy": True,
    "use_order_constraint": False,
}
SearchSpace(
    parameters=[
        RangeParameter(name="mu1_div_mu3", parameter_type=FLOAT, range=[0.
                                                    3333333333333333, 1.
                                                    6666666666666667]),
        RangeParameter(name="mu2_div_mu3", parameter_type=FLOAT, range=[0.
                                                    3333333333333333, 1.
                                                    6666666666666667]),
        RangeParameter(name="std1", parameter_type=FLOAT, range=[0.1, 1.0]),
        RangeParameter(name="std2", parameter_type=FLOAT, range=[0.1, 1.0]),
        RangeParameter(name="std3", parameter_type=FLOAT, range=[0.1, 1.0]),
        RangeParameter(name="comp1", parameter_type=FLOAT, range=[0.0, 1.0]),
        RangeParameter(name="comp2", parameter_type=FLOAT, range=[0.0, 1.0]),
    ],
    parameter_constraints=[ParameterConstraint(1.0 * comp1 + 1.0 * comp2 <= 1.0)],
)
```

S8.3. Size and Order

```
{
    "remove_scaling_degeneracy": True,
    "remove_composition_degeneracy": False,
    "use_order_constraint": True,
}
SearchSpace(
    parameters=[
        RangeParameter(name="mu1_div_mu3", parameter_type=FLOAT, range=[0.
                                                    3333333333333333, 1.
                                                    6666666666666667]),
        RangeParameter(name="mu2_div_mu3", parameter_type=FLOAT, range=[0.
                                                    3333333333333333, 1.
                                                    6666666666666667]),
        RangeParameter(name="std1", parameter_type=FLOAT, range=[0.1, 1.0]),
        RangeParameter(name="std2", parameter_type=FLOAT, range=[0.1, 1.0]),
        RangeParameter(name="std3", parameter_type=FLOAT, range=[0.1, 1.0]),
        RangeParameter(name="comp1", parameter_type=FLOAT, range=[0.0, 1.0]),
        RangeParameter(name="comp2", parameter_type=FLOAT, range=[0.0, 1.0]),
        RangeParameter(name="comp3", parameter_type=FLOAT, range=[0.0, 1.0]),
    ],
    parameter_constraints=[OrderConstraint(std1 <= std2)],
)
```

S8.4. Size

```
{
    "remove_scaling_degeneracy": True,
    "remove_composition_degeneracy": False,
    "use_order_constraint": False,
}
SearchSpace(
    parameters=[
        RangeParameter(name="mu1_div_mu3", parameter_type=FLOAT, range=[0.
                                                    3333333333333333, 1.
                                                    6666666666666667]),
        RangeParameter(name="mu2_div_mu3", parameter_type=FLOAT, range=[0.
                                                    3333333333333333, 1.
                                                    6666666666666667]),
        RangeParameter(name="std1", parameter_type=FLOAT, range=[0.1, 1.0]),
        RangeParameter(name="std2", parameter_type=FLOAT, range=[0.1, 1.0]),
        RangeParameter(name="std3", parameter_type=FLOAT, range=[0.1, 1.0]),
        RangeParameter(name="comp1", parameter_type=FLOAT, range=[0.0, 1.0]),
        RangeParameter(name="comp2", parameter_type=FLOAT, range=[0.0, 1.0]),
        RangeParameter(name="comp3", parameter_type=FLOAT, range=[0.0, 1.0]),
    ],
    parameter_constraints=[],
)
```

S8.5. Composition and Order

```
{
    "remove_scaling_degeneracy": False,
    "remove_composition_degeneracy": True,
    "use_order_constraint": True,
}
SearchSpace(
    parameters=[
        RangeParameter(name="mu1", parameter_type=FLOAT, range=[1.0, 5.0]),
        RangeParameter(name="mu2", parameter_type=FLOAT, range=[1.0, 5.0]),
        RangeParameter(name="mu3", parameter_type=FLOAT, range=[1.0, 5.0]),
        RangeParameter(name="std1", parameter_type=FLOAT, range=[0.1, 1.0]),
        RangeParameter(name="std2", parameter_type=FLOAT, range=[0.1, 1.0]),
        RangeParameter(name="std3", parameter_type=FLOAT, range=[0.1, 1.0]),
        RangeParameter(name="comp1", parameter_type=FLOAT, range=[0.0, 1.0]),
        RangeParameter(name="comp2", parameter_type=FLOAT, range=[0.0, 1.0]),
    ],
    parameter_constraints=[
        ParameterConstraint(1.0 * comp1 + 1.0 * comp2 <= 1.0),
        OrderConstraint(std1 <= std2),
        OrderConstraint(std2 <= std3),
    ],
)
```

S8.6. Composition

```
{
    "remove_scaling_degeneracy": False,
    "remove_composition_degeneracy": True,
    "use_order_constraint": False,
}
SearchSpace(
    parameters=[
        RangeParameter(name="mu1", parameter_type=FLOAT, range=[1.0, 5.0]),
        RangeParameter(name="mu2", parameter_type=FLOAT, range=[1.0, 5.0]),
        RangeParameter(name="mu3", parameter_type=FLOAT, range=[1.0, 5.0]),
        RangeParameter(name="std1", parameter_type=FLOAT, range=[0.1, 1.0]),
        RangeParameter(name="std2", parameter_type=FLOAT, range=[0.1, 1.0]),
        RangeParameter(name="std3", parameter_type=FLOAT, range=[0.1, 1.0]),
        RangeParameter(name="comp1", parameter_type=FLOAT, range=[0.0, 1.0]),
        RangeParameter(name="comp2", parameter_type=FLOAT, range=[0.0, 1.0]),
    ],
parameter_constraints=[ParameterConstraint(1.0 * comp1 + 1.0 * comp2 <= 1.0)],
)
```

S8.7. Order

```
{
    "remove_scaling_degeneracy": False,
    "remove_composition_degeneracy": False,
    "use_order_constraint": True,
}
SearchSpace(
    parameters=[
        RangeParameter(name="mu1", parameter_type=FLOAT, range=[1.0, 5.0]),
        RangeParameter(name="mu2", parameter_type=FLOAT, range=[1.0, 5.0]),
        RangeParameter(name="mu3", parameter_type=FLOAT, range=[1.0, 5.0]),
        RangeParameter(name="std1", parameter_type=FLOAT, range=[0.1, 1.0]),
        RangeParameter(name="std2", parameter_type=FLOAT, range=[0.1, 1.0]),
        RangeParameter(name="std3", parameter_type=FLOAT, range=[0.1, 1.0]),
        RangeParameter(name="comp1", parameter_type=FLOAT, range=[0.0, 1.0]),
        RangeParameter(name="comp2", parameter_type=FLOAT, range=[0.0, 1.0]),
        RangeParameter(name="comp3", parameter_type=FLOAT, range=[0.0, 1.0]),
    ],
    parameter_constraints=[
        OrderConstraint(std1 <= std2),
        OrderConstraint(std2 <= std3),
    ],
)
```

S8.8. Bounds-only

```
{
    "remove_scaling_degeneracy": False,
    "remove_composition_degeneracy": False,
    "use_order_constraint": False,
}
SearchSpace(
    parameters=[
        RangeParameter(name="mu1", parameter_type=FLOAT, range=[1.0, 5.0]),
        RangeParameter(name="mu2", parameter_type=FLOAT, range=[1.0, 5.0]),
        RangeParameter(name="mu3", parameter_type=FLOAT, range=[1.0, 5.0]),
        RangeParameter(name="std1", parameter_type=FLOAT, range=[0.1, 1.0]),
        RangeParameter(name="std2", parameter_type=FLOAT, range=[0.1, 1.0]),
        RangeParameter(name="std3", parameter_type=FLOAT, range=[0.1, 1.0]),
        RangeParameter(name="comp1", parameter_type=FLOAT, range=[0.0, 1.0]),
        RangeParameter(name="comp2", parameter_type=FLOAT, range=[0.0, 1.0]),
        RangeParameter(name="comp3", parameter_type=FLOAT, range=[0.0, 1.0]),
    ],
    parameter_constraints=[],
)
```

References

[1] 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.

[2] A. Huamán, M. Quintana, J. Rodriguez, W. Estrada, Dye-Sensitized Solar Cells Based on TiO₂ Nanoparticles Modified by Wet Milling, EPE 06 (2014) 473–480. doi:10.4236/epe.2014.613040.

[3] S. Pratsinis, Milling & Analysis of Particles, 2018.

[4] D. Stamatis, Z. Jiang, V. K. Hoffmann, M. Schoenitz, E. L. Dreizin, Fully Dense, Aluminum-Rich Al-CuO Nanocomposite Powders for Energetic Formulations, Combustion Science and Technology 181 (2008) 97–116. doi:10.1080/00102200802363294.