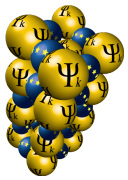

Ψ_k **Scientific Highlight Of The Month**

No. 155 **October 2023**

CELL: a Python package for cluster expansion with a focus on complex alloys

Santiago Rigamonti*, Maria Troppenz, Martin Kuban, Axel Huebner, and Claudia Draxl

Humboldt-Universität zu Berlin, Institut für Physik and IRIS Adlershof, 12489 Berlin, Germany

Abstract

We present the Python package CELL, which provides a modular approach to the cluster expansion (CE) method. CELL can treat a wide variety of substitutional systems, including one-, two-, and three-dimensional alloys, in a general multi-component and multi-sublattice framework. It is capable of dealing with complex materials comprising several atoms in their *parent lattice*. CELL uses state-of-the-art techniques for the construction of training data sets, model selection, and finite-temperature simulations. The user interface consists of well-documented Python classes and modules (http://sol.physik.hu-berlin.de/cell/). CELL also provides visualization utilities and can be interfaced with virtually any *ab initio* package, total-energy codes based on interatomic potentials, and more. The usage and capabilities of CELL are illustrated by a number of examples, comprising a Cu-Pt surface alloy with oxygen adsorption, featuring two coupled binary sublattices, and the thermodynamic analysis of its order-disorder transition; the demixing transition and lattice-constant bowing of the Si-Ge alloy; and an iterative CE approach for a complex clathrate compound with a parent lattice consisting of 54 atoms.

1 Introduction

Typical quests in materials science, as for instance finding stable compositions of an alloy and its properties, or determining the conditions for molecular adsorption on a surface, involve computations of a large number of atomic configurations on a well-defined lattice. Ideally, one would perform these computations using first-principles methods, *e.g.*, density-functional theory (DFT), however, due to the combinatorial explosion of the number of configurations with system size, a direct *ab initio* approach is in many cases out of reach. In this context, the cluster expansion (CE) method [1, 2] can be used to achieve a physical description of materials with essentially *ab initio* accuracy at reasonable computational cost. In this method, the configuration-dependent properties of the material are computed by means of generalized Ising-like models parametrized with *ab initio* data. Thus, calculations of a huge number of

*srigamonti@physik.hu-berlin.de

atomic configurations with large supercells become feasible, bridging length scales and enabling a statistical-thermodynamics description.

Despite these capabilities, systems of technological interest are often still too complex. Dealing with complexity requires adequate code, able to account for multi-component settings, multiple sublattices, large parent lattices, surfaces and interfaces, and more. Moreover, CE modeling also entails tasks like *learning from data*, including the creation of data sets for model training and the evaluation and selection of models. In this context, the application of modern artificial intelligence (AI) techniques to CE model construction is possible and desirable. In this work, we present the CE Python [3] package `CELL` [4] that provides a modular approach to cluster expansion, fulfilling all the needs mentioned above. `CELL` allows the set-up of systems with an arbitrary number of substituent species types and an arbitrary number of sublattices. Here, a sublattice is a subset of crystal sites whose composition can differ from other sites. The size of the parent lattice, *i.e.*, the number of atoms comprising the primitive unit cell of the pristine (non-substituted) material, can be arbitrarily large, so that `CELL` can readily deal with complex alloys having several atoms in the primitive unit cell. One, two, and three dimensions can be handled, which allows, for instance, the study of surface alloys.

The construction of CE models involves fitting to, *e.g.*, *ab initio* data sets. Since these data are often very costly to compute, `CELL` implements a number of structure selection strategies aimed at rationalizing data sets for optimal model training. These include special quasirandom structures [5] and variance-reduction schemes [6, 7]. `CELL` is written in Python [8], and provides a common interface to the various estimators from the machine-learning Python library `scikit-learn` [9, 10], and to native estimators, *i.e.*, those coded inside the `CELL` package. Its modularity allows for the *ad hoc* design and implementation of AI strategies. At the core of the representation of crystal structures, the `Atoms` object of the Atomic Simulation Enviroment (ASE) [11] is employed, giving access to all the advantages of ASE, as for instance the construction of structure databases and the interface to a large number of *ab initio* DFT codes. `CELL` provides state-of-the-art tools for performing thermodynamic analysis of materials. To this extent, a module for the calculation of the configurational density of states with the Wang-Landau [12] method is available. This module can be executed in parallel, which paves the way for performing simulations of very large supercells.

The paper is organized as follows: Sec. 2 gives a general introduction to the CE method; Sec. 3 and Sec. 4 present the structure of the code for model construction and thermodynamical analysis, respectively, along with an illustration of its most important features through a complex surface system consisting of a Pt/Cu(111) surface alloy with atomic O adsorption. Finally, the application to SiGe, and the complex thermoelectric clathrate compound $Ba_8Al_xSi_{46-x}$, with 54 atoms in the primitive cell, are presented in Sec. 5. Conclusions are given in Sec. 6.

2 Cluster expansion method

In this section, we present an overview of the cluster expansion formalism for the multi-component and multi-sublattice case [13, 14]. Additionally, formal aspects related to the application of AI techniques to CE model construction are introduced.

2.1 Simple CE of a binary alloy

Figure 1 shows a schema of a binary substitutional system where every site of a square lattice can be occupied by one of two species, indicated by blue and red color, respectively. The goal is to find the relation between the configuration of the substituent species (red) and the physical properties of the system, for instance the total energy. This question could be answered by performing numerically costly first-principles calculations for every atomic arrangement of the lattice. Though quite convenient, this simple strategy turns out to be impractical since the number of configurations scales as 2^N, with N being the number of lattice sites (for the system of Fig. 1, the number of configurations would be of the order of 7×10^4, without considering symmetries). However, by making a few *reasonable* assumptions, we can calculate the energy of an arbitrary configuration as in the left side of Fig. 1 in a very efficient way. The only input needed is the energies of a small number of well-chosen configurations.

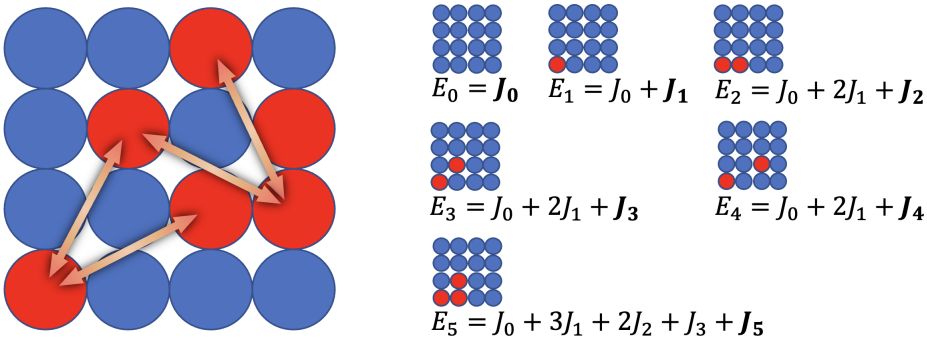

Figure 1: Example of a two dimensional binary system. The properties (*e.g.*, the energy E) of an arbitrary configuration (left) can be approximately computed in terms of a superposition of effective n-body interactions (right).

We start by considering *effective* n-body interactions and assume that (i) the 1-body interactions are larger than the 2-body interactions, these being larger than the 3-body interactions, and so on; (ii) the interactions decrease with distance; and (iii) the total energy can be written as a linear superposition of n-body interactions. Next, we evaluate the *ab initio* energies E_0 to E_5 of the configurations depicted on the right side of Fig. 1. According to the assumptions made above, these contain *important* interactions that are labeled with J_0 to J_5: J_0 represents the energy E_0 of the pristine structure (all circles blue). J_1 accounts for the change in energy of the system when a "blue" species is replaced by a "red" one, and can be obtained by subtracting J_0 from the computed E_1. J_2 to J_4 represent 2-body interactions of increasing distance. They can be determined by making use of assumption (iii). For instance, the calculated value of E_2 is the addition of the energy of the pristine structure (J_0), plus two "blue"-to-"red" susbstitutions ($2J_1$), plus an additional term (J_2) that embodies a 2-body interaction and can be estimated by the difference $E_2 - (J_0 + 2J_1)$. Finally, J_5 represents a 3-body interaction that can be estimated analogous to the 2-body interactions. With these so-called effective cluster interactions $J_{i=0-5}$, we can now predict the energy of the configuration on the left side of Fig. 1 as

$$\hat{E} = J_0 + 6J_1 + 2J_2 + 4J_3 + 4J_4 + 1J_5. \tag{1}$$

If assumptions (i) to (iii) are true, then the predicted energy value $\hat{E}$ will be close to the *ab*

initio energy E (henceforth predicted values will be denoted by a "hat" symbol). The integer numbers multiplying the interactions J_i tell how many times the corresponding pattern or *cluster* is present in a given structure. For instance, it is 4 for the interaction J_4, as indicated by the four arrows in Fig. 1.

The naive approach just exposed has two important shortcomings. First, the assumtions (i) and (ii) cannot be expected to be true for all material properties. Second, it cannot be easily adapted to more complex situations as, *e.g.*, shown in Fig. 2. In this case, we have a parent lattice (top left) with different *sublattice types* assigned to the lattice points. These are indicated by white, striped, and gridded patterns. Different numbers of substitutions and species types can occupy each sublattice, as indicated by the colors (top right). Large supercells can be constructed by periodic repetitions of the parent lattice (bottom left). These allow for the construction of configurations compatible with the parent lattice, such as fully disordered, ordered, or partially (dis)ordered structures as shown in the bottom right. Analogous situations are frequently encountered, *e.g.*, in complex bulk or surface alloys.

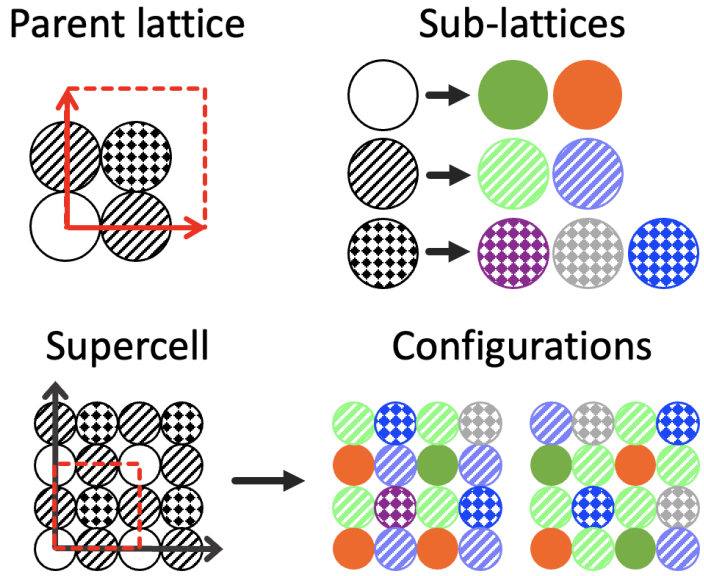

Figure 2: Example of a two-dimensional system with a multi-composition and multi-sublattice setup. The parent lattice with four sites contains three different sub-lattices (top left). Each of them can host different numbers and kinds of atoms, as indicated by colors (top right). Supercells (bottom left) based on this parent lattice (dashed box), serve as "blue-print" for the construction of configurations (bottom right) compatible with the sublattice definition.

In the following, we will explain how to treat this general case. The formulation does not make use of assumptions (i) and (ii) but allows for the application of AI methods to identify property-specifc interactions.

2.2 General CE formalism

We consider a crystal lattice with atomic positions $\mathbf{R}_i$, $i = 1, ..., N$. Every position $\mathbf{R}_i$ can host any of M_i atoms of different type. An arbitrary arrangement of species in the lattice can be represented by vector $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, ..., \sigma_N)$, with σ_i being integer numbers between 0 and $M_i - 1$,

indicating the species at position $\mathbf{R}_i$. The pristine crystal is defined as the configuration with $\sigma_i = 0$ for all i.

A physical property that depends on the configuration can be represented by a function $P(\boldsymbol{\sigma})$. It can be shown [2, 14] that, in the discrete space spanned by the vectors $\boldsymbol{\sigma}$, there exist complete and orthonormal sets of basis functions $\Gamma_{\boldsymbol{\alpha}}(\boldsymbol{\sigma})$, called *cluster functions*, in terms of which $P(\boldsymbol{\sigma})$ can be expanded:

$$P(\boldsymbol{\sigma}) = \sum_{\boldsymbol{\alpha}} J_{\boldsymbol{\alpha}} \Gamma_{\boldsymbol{\alpha}}(\boldsymbol{\sigma}). \tag{2}$$

The real numbers $J_{\boldsymbol{\alpha}}$ are the expansion coefficients and $\boldsymbol{\alpha}$ stands for a vector of components $\alpha_i \in \{0, 1, ...M_i - 1\}$, $i = 1, ..., N$. The cluster functions fulfil the orthonormality condition $\langle \Gamma_{\boldsymbol{\alpha}}, \Gamma_{\boldsymbol{\beta}} \rangle = \delta_{\boldsymbol{\alpha}\boldsymbol{\beta}}$. Here, $\delta_{\boldsymbol{\alpha}\boldsymbol{\beta}} = \prod_{i=1}^{N} \delta_{\alpha_i \beta_i}$, with $\delta_{\alpha_i \beta_i}$ being the Kronecker delta, and the inner product is defined by $\langle f, g \rangle = \sum_{\boldsymbol{\sigma}} f(\boldsymbol{\sigma}) g(\boldsymbol{\sigma}) / \prod_{i=1}^{N} M_i$, with f and g arbitrary functions in configuration space and the sum running over all possible configurations $\boldsymbol{\sigma}$ of the system. The cluster functions can be constructed as follows [2, 14]:

$$\Gamma_{\boldsymbol{\alpha}}(\boldsymbol{\sigma}) = \prod_{i=1}^{N} \gamma_{M_i, \alpha_i}(\sigma_i), \tag{3}$$

with the functions $\gamma_{M_i, \alpha_i}(\sigma_i)$ forming a real, orthonormal basis in the discrete (and finite) domain $\sigma_i = 0, 1, ..., (M_i - 1)$. Usual choices for this basis include discrete Chebyshev polynomials and trigonometric functions [2, 14, 15].

Without loss of generality, one can choose the constant function $\gamma_{M,0}(\sigma) = 1$ for $\alpha = 0$. Thus, the product in Eq. (3) can be restricted to the indices $\alpha_i \neq 0$. Accordingly, a cluster function is defined by indicating the set $\{(i, \alpha_i) | \alpha_i \neq 0\}$. The special case where all α_i are 0, corresponds to the *empty cluster* function, $\Gamma_{\emptyset}(\boldsymbol{\sigma}) = 1$. There are M_i basis functions γ at a site i. Obviously, if two clusters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are related by a symmetry operation of the parent lattice, then their expansion coefficients are equal, *i.e.*, $J_{\boldsymbol{\alpha}} = J_{\boldsymbol{\beta}}$. It should be noted that the sublattice types also determine the symmetry. Hence, the sum in Eq. (2) can be split into a sum over symmetrically inequivalent (s.i.) clusters and a sum over symmetrically equivalent ones:

$$P(\boldsymbol{\sigma}) = \sum_{\boldsymbol{\alpha}}^{\text{s.i.}} \mathcal{M}_{\boldsymbol{\alpha}} J_{\boldsymbol{\alpha}} X_{\boldsymbol{\alpha}}(\boldsymbol{\sigma}). \tag{4}$$

Here, the *cluster correlation function* $X_{\boldsymbol{\alpha}}(\boldsymbol{\sigma})$

$$X_{\boldsymbol{\alpha}}(\boldsymbol{\sigma}) = \frac{1}{\mathcal{M}_{\boldsymbol{\alpha}}} \sum_{\boldsymbol{\beta} \in \mathcal{O}(\boldsymbol{\alpha})} \Gamma_{\boldsymbol{\beta}}(\boldsymbol{\sigma}). \tag{5}$$

is the average of the cluster functions in the set of clusters symmetrically equivalent to $\boldsymbol{\alpha}$. This set, of size $\mathcal{M}_{\boldsymbol{\alpha}}$, is called the *orbit* of cluster $\boldsymbol{\alpha}$, and we denote it by $\mathcal{O}(\boldsymbol{\alpha})$.

Models are often constructed for an intensive property, as for instance the energy per unit cell. In such a case, $\mathcal{M}_{\boldsymbol{\alpha}}$ in Eq. (4) can be replaced by $m_{\boldsymbol{\alpha}} = \mathcal{M}_{\boldsymbol{\alpha}} V_{pc} / V_{sc}$, with V_{sc} and V_{pc} being the super- and the parent-cell volume, respectively. The integers $m_{\boldsymbol{\alpha}}$ are also called cluster multiplicities. In the simple binary case of Fig. 1, we have $M_i = 2$. If we choose the basis $\gamma_{20}(\sigma) = 1$, $\gamma_{21}(\sigma) = \sigma$, $\sigma = 0, 1$ *, it is easy to verify that, for the structure represented on

*This basis is not orthonormal, however, this is not problematic for the present example.

the left side of Fig. 1, the cluster correlations for clusters 1 to 5 on the right are, respectively, $X = 6/16$, $2/32$, $4/32$, $4/64$, and $1/64$, with the denominators being the corresponding values of $\mathcal{M}$. Thus, the values of $\mathcal{M}X = 6$, 2, 4, 4, 1 agree with the coefficients for J_1 to J_5 in Eq. (1).

The number of clusters in the expansion of Eq. (4) is in principle infinite, and the expansion coefficients $J_{\boldsymbol{\alpha}}$ (called effective cluster interactions, ECIs in short), are still undetermined. For practical applications, though, it is necessary to cut off the cluster basis, keeping only the most *relevant* clusters in Eq. (4). One also has to determine the ECIs that lead to accurate predictions of the property of interest. These are the main tasks to be addressed in the construction of CE models. Below, we briefly explain how this problem is tackled by using AI techniques.

2.3 CE as a data-analytics problem

To build a CE model of a material property of interest, we need (i) a set of structures $\mathcal{S} = \{\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2, ..., \boldsymbol{\sigma}_{N_s}\}$, (ii) the corresponding calculated properties $\boldsymbol{P}^\top = (P_1, P_2, ..., P_{N_s})$ with $P_i = P(\boldsymbol{\sigma}_i)$, and (iii) a set of clusters $\mathcal{C} = \{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, ..., \boldsymbol{\alpha}_{N_c}\}$. Then, we build a matrix $\boldsymbol{X}$ of cluster correlations, with elements $X_{ij} = X_{\boldsymbol{\alpha}_j}(\boldsymbol{\sigma}_i)$, such that a column j in the matrix represents a cluster, while a row i represents a structure. By using Eq. (4) with the sum on clusters limited to the set $\mathcal{C}$, we can write, for an intensive property

$$\hat{\boldsymbol{P}} = \boldsymbol{X}\boldsymbol{\mathcal{J}}, \tag{6}$$

where $\boldsymbol{\mathcal{J}}$ is a column vector with elements $\mathcal{J}_i = m_{\boldsymbol{\alpha}_i} J_{\boldsymbol{\alpha}_i}$, $i = 1, ..., N_c$. Optimal cluster interactions $\boldsymbol{\mathcal{J}}$ can be found by minimizing a cost function:

$$\boldsymbol{\mathcal{J}} = \underset{\boldsymbol{\mathcal{J}}^*}{\operatorname{argmin}} \left[\|\boldsymbol{X}\boldsymbol{\mathcal{J}}^* - \boldsymbol{P}\|_2^2 + \phi(\boldsymbol{\mathcal{J}}^*) \right]. \tag{7}$$

Here, we use the standard definition of the ℓ_p-norm of a vector $\boldsymbol{x} = (x_1, ..., x_n)$, namely $\|\boldsymbol{x}\|_p = (\sum_{i=1}^n |x_i|^p)^{1/p}$. (In the particular case, $p = 0$, $\|\boldsymbol{x}\|_0$ is defined as the number of non-zero components of $\boldsymbol{x}$.) The first term inside the square brackets is the mean squared error (MSE) of the predictions, and the second term, $\phi(\boldsymbol{\mathcal{J}}^*)$, is a penalization or regularization term. The latter can be used for different purposes. For instance, if there are no linearly dependent columns in $\boldsymbol{X}$ and $N_c \leq N_s$, *i.e.*, the number of clusters is smaller than the number of structures in the training set $\mathcal{S}$, optimal cluster interactions $\boldsymbol{\mathcal{J}}$ can found by directly minimizing the MSE of the predictions ($\phi = 0$), with the solution $\boldsymbol{\mathcal{J}} = \left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{P}$.

In typical applications, data are scarce, and one wants to find the relevant interactions out of a large set of clusters, *i.e.*, solve an underdetermined problem with $N_c > N_s$. In this case, the Gram matrix $\boldsymbol{X}^\top \boldsymbol{X}$ cannot be inverted. Thus, the optimization problem posed by Eq. (7) must be regularized. The easiest way to achieve this is by choosing $\phi(\boldsymbol{\mathcal{J}}) = \lambda \|\boldsymbol{\mathcal{J}}\|_2^2$, with the hyperparameter $\lambda \in \Re$. In this case, the solution is $\boldsymbol{\mathcal{J}} = \left(\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{P}$ [16]. Besides making the problem solvable, this selection penalizes large ECI values, leading to solutions with increasingly small interactions for increasing λ. To obtain sparse models in which the relevant interactions are represented by a small number of clusters, compressed-sensing techniques [17, 18] may be employed. In compressed sensing, one seeks penalization terms in Eq. (7) that promote sparsity, *i.e.*, giving solutions with many interactions $\mathcal{J}_i$ being exactly zero. Common

penalization terms fulfilling this requirement, are $\phi = \lambda\|\boldsymbol{\mathcal{J}}\|_0$ or $\phi = \lambda\|\boldsymbol{\mathcal{J}}\|_1$. The first choice penalizes solutions with a large number of non-zero parameters by using the ℓ_0-norm, thus producing sparse models. The associated minimization problem is NP-hard [19], *i.e.*, its solution cannot be found in polynomial time, and exact solutions can only be computed for rather small clusters pools. Using the second choice (also called the *Manhattan norm*), leads to the least-absolute-shrinkage-and-selection-operator (LASSO) approach [20], that represents a convex optimiztion problem, and efficient algorithms exist for its solution. Under certain conditions [21], the solutions found with LASSO may approximate well the solutions found using the ℓ_0 norm.

Once the optimal $\boldsymbol{\mathcal{J}}$ is found by solving Eq. (7), one can use Eq. (4) to perform property predictions for arbitrary configurations $\boldsymbol{\sigma}$.

3 Cluster expansion with `CELL`

To demonstrate the construction of CE models with `CELL`, we build one for the energy of formation of a complex surface system, consisting of a Pt/Cu(111) surface alloy with atomic O adsorption, as shown in Fig. 3. In this example, both O adsorption and Pt-Cu alloying phenomena are simultaneously accounted for. Platinum and copper form a two-dimensional (2D) alloy in the top-most atomic layer of Cu(111), creating disordered as well as ordered 2D surface patterns (*e.g.*, p(2×2) and $\sqrt{3} \times \sqrt{3}$ R 30° reconstructions) [22]. Atomic oxygen adsorbs preferentially on hollow fcc sites, both on the pristine Cu(111) [23] and Pt(111) [23, 24] surfaces. These facts define the parent lattice of our system.

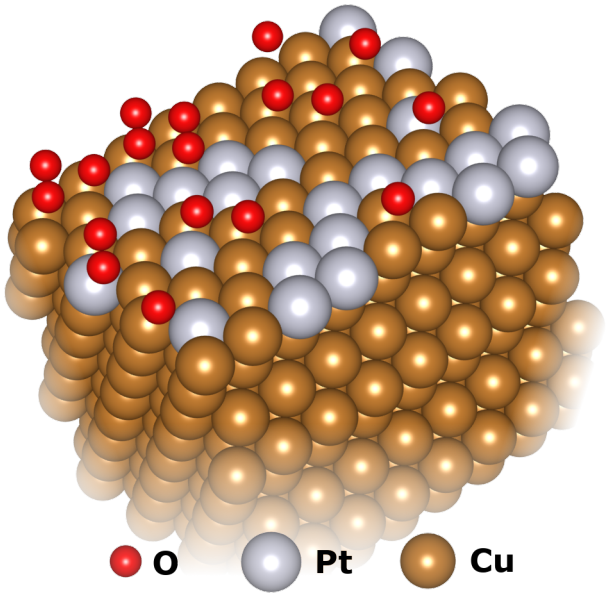

Figure 3: Pt/Cu(111) surface alloy with atomic O adsorption.

To avoid the generation of numerically costly DFT data sets, the total energies in the present example are calculated with the effective medium theory (EMT) calculator from ASE [11]. While the EMT potentials for Pt and Cu are quite realistic [25], this is not the case for oxygen. Since, however, the main purpose of this section is to showcase the construction of a CE model and its use and not to capture the actual physical details of the system, we proceed in this way. The use

of toy total-energy models is a common practice to test CE methods (see *e.g.*, Refs. [18, 26, 27]).

All the code listings shown in this section are written in Python 3 [3] and can be run interactively as Jupyter notebooks [28] (see Sec. 7).

3.1 Generation of structures

The basic building block for the generation of structures for the CE is the parent lattice. It comprises the definition of the primitive cell of the pristine crystal and the species that can possibly occupy any crystal site. In `CELL`, a parent lattice is embodied by the `ParentLattice` class. It admits the definition of a multi-sublattice and multi-composition framework, as demonstrated in Listing 1. In lines 1 to 5 of the listing, the Atomic Simulation Environment (ASE) [11] is used to create an `Atoms` object representing the primitive cell of an fcc (111) Cu slab with three atomic layers and vacancy sites on hollow fcc positions (vacancies are indicated with the character `X`). This `Atoms` object is assigned to the variable `pristine`, as it represents the pristine primitive cell, hosting no substituents. In line 8, the `ParentLattice` class of `CELL` is loaded [†], and a `ParentLattice` object is initialized in line 11 and assigned to the variable `p_lat`. The initialization takes two arguments: the primitive cell (variable `pristine`), and a list of possible species that every crystal site can host. The latter is defined in the list `symbols` in line 10 of the listing: The first two layers may only contain Cu (thus behaving as *spectator* atoms [6], *i.e.*, they determine the symmetry but are not part of any cluster), while the Cu atoms on the top layer may be substituted by Pt, and the remaining vacancy sites (X) can be substituted by oxygen. Note that the order of this list must correspond to the atomic arrangement in the `Atoms` object `pristine`. The latter could be inquired with the method `pristine.get_positions()`.

```
1  from ase.build import fcc111, add_adsorbate
2
3  pristine = fcc111('Cu', a=3.59, size=(1,1,3)) # 3-atomic-layer Cu(111) slab
4  add_adsorbate(pristine,'X',1.7,position='fcc') # Hollow fcc vacancy site
5  pristine.center(vacuum=10.0, axis=2) # add vacuum along z-axis
6
7  # Note: CELL is imported with the name "clusterx"
8  from clusterx.parent_lattice import ParentLattice
9
10 symbols = [['Cu'],['Cu'],['Cu','Pt'],['X','O']]
11 p_lat = ParentLattice(pristine, symbols=symbols)
12 p_lat.print_sublattice_types()
```

Listing 1: Creation of a two-dimensional multi-composition multi-sublattice `ParentLattice` object.

Upon execution of this code, the output, as shown in Fig. 4, is displayed, indicating that three sublattices were created: two different binary sublattices (assigned sublattice types 0 and 2) and one spectator sublattice (sublattice type 1). In this way, a full multi-component multi-sublattice framework can be generated. `CELL` does not restrict the number of n-ary (*i.e.*, unary, binary,

[†]Note that the `CELL` package is called and imported in Python code with the name `clusterx`.

ternary, ...) sublattices that can be defined in this way.

We can now build a supercell and visualize it, as shown in Listing 2. In line 2 of the listing, we use the `SuperCell` class of `CELL` to create a `SuperCell` object, based on the parent lattice created in Listing 1. The lattice coordinates of the supercell vectors are (4,0) and (-2,4), referring to the hexagonal unit vectors of the Cu(111) surface (indicated by arrows in the left panel of Fig. 5). The call to the `juview` method (`CELL`'s plotting interface for Jupyter notebooks) in line 5 produces the image shown in Fig. 5. In this graphical representation of the `SuperCell`, the first image on the left depicts the pristine, non-substituted crystal, while the images on the right, represent the results of substituting one of the species as allowed in the definition of the parent lattice.

```
1 from clusterx.super_cell import SuperCell
2 scell = SuperCell(p_lat,[[4,0],[-2,4]])
3
4 from clusterx.visualization import juview
5 juview(scell)
```

Listing 2: Creation and visualization of a `SuperCell` object.

A supercell like the one depicted in Fig. 5, serves as a blueprint for the generation of structures. In Listing 3, we use the supercell to construct random structures and gather them in a `StructuresSet` object, which is created in line 2 of the listing. Objects of this class act as structure containers which have a database attached. They can be serialized, *e.g.*, in the form of JSON database files fully compatible with ASE's JSON databases.

```
1 from clusterx.structures_set import StructuresSet
2 sset = StructuresSet(p_lat)
3 for i in range(50):
4    rnd_str = scell.gen_random_structure()
5    sset.add_structure(rnd_str)
6 juview(sset,n=4)
```

Listing 3: Creation of a `StructuresSet` object containing 50 random structures.

In line 4, structure creation takes place by calling the `gen_random_structure()` method of the `SuperCell` class. This returns a `Structure` object. In `CELL`, `Structure` objects consist of a

```
+-------------------------------------------------------------------+
|            The structure consists of 3 sublattices                |
+-------------------------------------------------------------------+
| Sublattice type |     Chemical symbols     |  Atomic numbers    |
+-------------------------------------------------------------------+
|        0        |        ['X' 'O']         |       [0 8]        |
|        1        |          ['Cu']          |       [29]         |
|        2        |       ['Cu' 'Pt']        |      [29 78]       |
+-------------------------------------------------------------------+
```

Figure 4: Output of Listing 1, corresponding to a `ParentLattice` object with three sublattices.

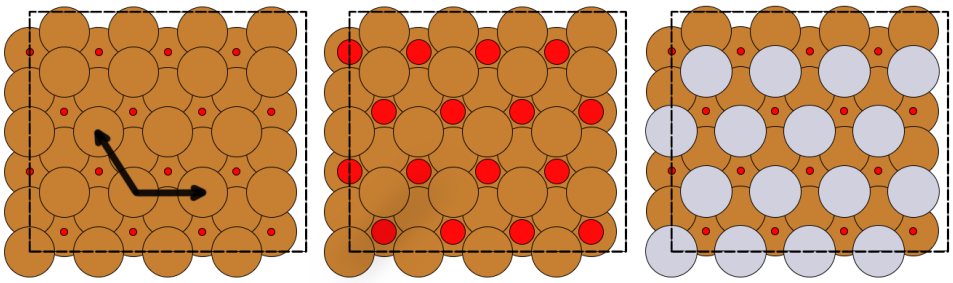

Figure 5: Graphical represention of a `SuperCell` object. Brown, small red, big red, and gray spheres represent copper, vacancies, oxygen, and platiunum atoms, respectively. From left to right: pristine supercell; full substitution of vacant hollow fcc sites by O atoms; full substitution of the top Cu layer by Pt atoms. Arrows indicate the unit cell vectors of the underlying parent lattice.

supercell augmented with a *decoration* array, which is a list indicating which species occupy the individual sites of the supercell. Thus, all relevant information like, *e.g.*, sublattice types, is provided. Less memory-consuming representations consisting of only the decoration array, are available for tasks such as Monte Carlo simulations (see Sec. 4.1).

The 50 generated structures are added to `sset` in line 5 of Listing 3 by using the `add_structure()` method of the `StructuresSet` class. Finally, four of the generated structures are displayed by calling the `juview()` method in line 6, with the result as shown in Fig. 6.

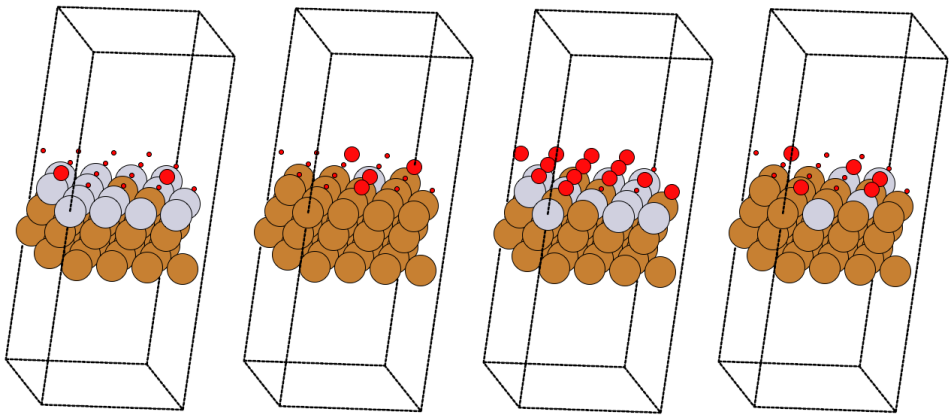

Figure 6: Four simulation cells of random structures belonging to the `StructuresSet` object of Listing 3. The color code is the same as in Fig. 5.

In summary, the construction and collection of structures in `CELL` takes place in four classes, three of them related by inheritance as shown in Fig. 7: The `ParentLattice` class inherits from ASE's `Atoms` class and is supplied with a multi-composition-multi-sublattice framework; a `SuperCell` is an enlarged `ParentLattice`; a `Structure` is a `Supercell` *decorated* with a specific configuration of substituent species. These classes are equipped with a lot of useful functionality, either through methods inherited from ASE's `Atoms` class, or from methods native to `CELL`. The latter are documented in Ref. [4]. `StructuresSet` objects allow, *e.g.*, for the union of sets through the "+" operator, serialization, aggregation of properties, and more.

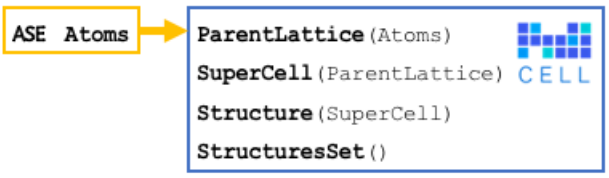

Figure 7: Inheritance map of `CELL`'s classes for structure construction and collection. Bold-face words indicate class names, and parenthesis indicate inheritance. Yellow box: `Atoms` class from ASE, blue box: classes belonging to `CELL`.

3.2 Calculation of configuration-dependent properties

Now that we have a set of structures, the next step in a standard CE workflow is to perform *ab initio* computations of the properties that we want to predict with a CE model. A common property to be modeled is the total internal energy of the alloy, $E(\boldsymbol{\sigma})$. With `CELL`, its computation can be easily achieved through the method `calculate_property()` of the `StructuresSet` class. All *ab initio* codes supported by the calculators of ASE can be employed. This is exemplified in the following listing by using the effective-medium-theory (`EMT`) calculator:

```
1 from ase.calculators.emt import EMT
2 sset.set_calculator(EMT()) # Assign EMT() calculator to StructuresSet.
3
4 # The total energy of every structure in sset is evaluated.
5 sset.calculate_property(prop_name="e_tot")
```

Listing 4: Calculation of the total energy of every structure in a `StructuresSet` object with a calculator of ASE.

Here, in line 1, the `EMT` calculator is loaded and an instance of it is attached to the `StructuresSet` object in line 2. Finally, in line 5 the calculator is used to compute the total energy of every structure in the set. By using the `prop_name` argument, we assign the name `"e_tot"` to this property. This acts, one the one hand, as a label to later retrieve the property values and, on the other hand, for storage upon serialization in, *e.g.*, a JSON file, with the `serialize()` method of `StructuresSet`.

For a surface system as the one we study here, a physically meaningful quantity is the adsorption energy, E_{ads}, defined as:

$$E_{\text{ads}}(\boldsymbol{\sigma}) = \frac{1}{N}\left[E(\boldsymbol{\sigma}) - n_{\text{O}}\frac{1}{2}E_{\text{O}_2} - n_{\text{Pt}}(E_{\text{Pt,bulk}} - E_{\text{Cu,bulk}}) - E_{\text{Cu,surf.}}\right] \tag{8}$$

Here, N is the number of Cu-Pt sites in the top-most layer of the supercell, n_{O} (n_{Pt}) the number of O (Pt) atoms, E_{O_2} the energy of an O_2 molecule, $E_{\text{Pt (Cu),bulk}}$ the energy per atom of fcc Pt (Cu), and $E_{\text{Cu,surf.}}$ the total energy of the pristine Cu slab. Since structural relaxation can notably affect the relative stability of structures, it is important to build the CE model with energies $E(\boldsymbol{\sigma})$ corresponding to optimized structures. The required steps are implemented in a custom python function, `ads_energy` (see Sec. 7), which gets a `Structure` object as argument and returns E_{ads}. Finally, the method `calculate_property` is called as follows:

```
sset.calculate_property(prop_name="e_ads", prop_func=ads_energy)
```

Listing 5: Calculation of the absorption energy of every structure in a `StructuresSet` object by means of the custom function `ads_energy`.

In more detail, `ads_energy` performs a structure optimization using the BFGS method. Three constraints are applied here: (i) The bottom-most Cu layer is fixed, (ii) the top-most Pt-Cu layer is allowed to relax in the (x, y) plane only, (iii) the O positions may relax in the perpendicular direction, z only. With the latter two constraints, we avoid reconstructions that may become significant at large Pt concentrations (*e.g.*, a Pt atom moving out from the Pt-Cu layer, or triads of O atoms sitting at bridge positions). We do so here for the sake of simplicity, although the degree of complexity brought about by these reconstructions could still be accounted for with `CELL` by adding the corresponding sites in the parent-lattice definition.

After evaluating the energies, we can plot the result by using the `plot_property_vs_concentration` method from `CELL`'s visualization package:

```
from clusterx.visualization import plot_property_vs_concentration
plot_property_vs_concentration(sset, site_type=2, property_name ="e_ads")
```

Listing 6: Code for generating a plot of the *ab initio* adsorption energy as a function of the Pt concentration.

Here, we instruct `CELL` to plot the adsorption energy calculated before (labeled `e_ads`) versus the concentration of the type-2 sublattice (*i.e.*, the Pt concentration in the Cu-Pt surface alloy). The result is shown in Fig. 8.

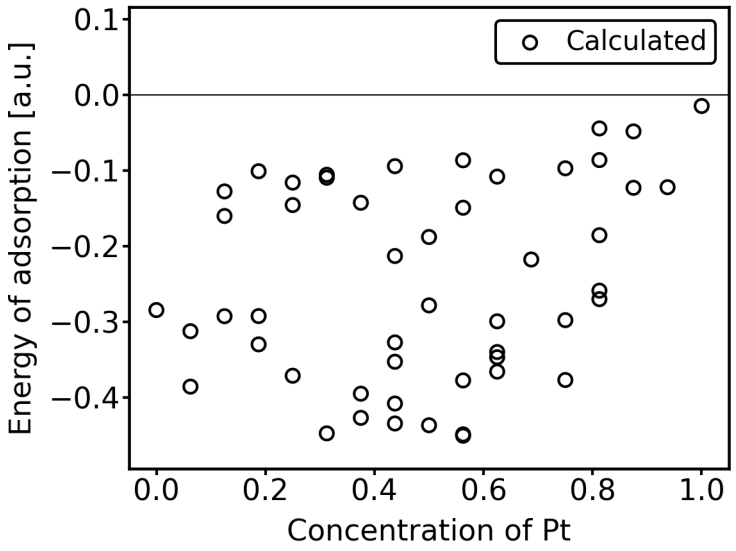

Figure 8: Energy of adsorption of 50 optimized random structures. The plot was created with Listing 6.

In this case, we have exemplified the calculation of properties by either using ASE's calculators

(Listing 4) or custom functions provided by the user (Listing 5). An approach consisting of the creation of folders containing input files for *ab initio* packages is also possible [4].

3.3 Clusters pool

Having the training data constructed, we have to define which cluster functions will constitute our basis set. This is done with the `ClustersPool` class of `CELL`, as shown in the following listing:

```
1 from clusterx.clusters.clusters_pool import ClustersPool
2 cpool = ClustersPool(p_lat,
3            npoints=[1,2,3,4],
4            radii=[0,-1,-1,4.3],
5            super_cell=scell)
6
7 cpool.serialize(db_name="cpool.json")
8 cpool.print_info()
```

Listing 7: Generation of a pool of clusters.

In line 1, we start by importing the `ClustersPool` class of `CELL`. Then, in lines 2-5, an instance of `ClustersPool` is created and assigned to the variable `cpool`. Clusters with 1 to 4 points are created, as indicated with the argument `npoints` in line 3. In line 4, the *radius* of every cluster, *i.e.*, the maximum distance between any of its sites, is specified. This is obviously 0 for the 1-point cluster. For clusters with 2 and 3 points, a negative radius (-1) is assigned. A negative number is used to indicate that all unique clusters compatible with the periodic boundary conditions of the supercell `scell`, specified in line 5 by the argument `super_cell`, are generated. For clusters with 4 points, we indicate a small radius of 4.3 Å. This selection is motivated by the notion that clusters with many points should mainly capture short-range-order effects. No general rule exists, nevertheless, and one should try different parameters for other properties or systems. The execution of line 7 serializes the `ClustersPool` to a JSON file. The created pool of clusters can then be conveniently visualized, for instance through the graphical user interface of ASE. Also using the `juview` function of `CELL` is possible, as explained before for the visualization of the `StructuresSet` (see Listing 3 and Fig. 6). In Fig. 9, all the generated 4-point clusters are shown. The execution of the last line generates the output shown in Fig. 10, listing index, number of points, radius, and multiplicity of every generated cluster.

3.4 Building CE models

The essential elements for the construction of a cluster expansion model are now available, namely, the training data contained in the `StructuresSet` object `sset` and the `ClustersPool` object `cpool`. With them, we can obtain the vector $\boldsymbol{P}$ and the input matrix $\boldsymbol{X}$ entering Eq. (7), as demonstrated in Listing 8.

Here, we start by creating a `CorrelationsCalculator` object in line 2. With this object, which is assigned to the variable `corrcal`, we can calculate the cluster correlations of Eq. (5) for any

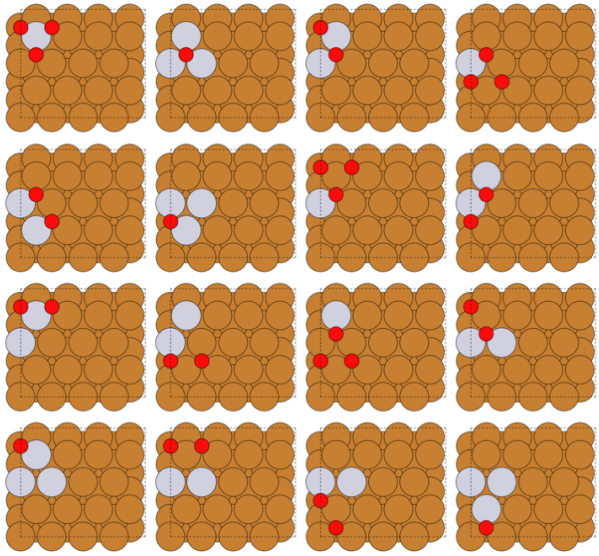

Figure 9: 4-point clusters with radii less than 4.3 Å gererated by `CELL`. The color code is not to be associated to atom types but to the index α_i of the basis functions $\gamma_{M_i,\alpha_i}(\sigma_i)$ of Eq. (3): For the top-layer sites (sublattice type 2, see Fig. 4), brown and gray correspond to $\alpha = 0$ and 1, respectively. For the fcc hollow sites (sublattice type 0), red circles correspond to $\alpha = 1$, while the remaining ones (not indicated) correspond to $\alpha = 0$.

structure based on the parent lattice `p_lat` and for all clusters in `cpool`. In this case, the calculator uses basis functions $\gamma_{M,\alpha}(\sigma)$ based on trigonometric functions, as indicated by the argument `"trigonometric"`. The full matrix $\boldsymbol{X}$ and the vector $\boldsymbol{P}$ are created in lines 5 and 8, respectively.

We now build a CE model by using the `EstimatorFactory` class of `CELL`. This class acts as an interface to all the estimators of the `scikit-learn` machine-learning Python library [9, 10] and to `CELL`'s native estimators.

In line 5 of Listing 9, a ridge-regression estimator from `scikit-learn` is created. This estimator solves Eq. (7) with an ℓ_2 regularization. The value `alpha=1e-8` indicates a small regularization parameter$\lambda = 10^{-8}$. Finally, a CE model is created and assigned to the variable `ce_model`

```
+-------------------------------------------------------------------+
|                        Clusters Pool Info                         |
+-------------------------------------------------------------------+
|      Index      |  Nr. of points  |     Radius      |  Multiplicity  |
+-------------------------------------------------------------------+
|        0        |        1        |      0.000      |        1       |
|        1        |        1        |      0.000      |        1       |
|        2        |        2        |      2.245      |        3       |
|       ...       |       ...       |       ...       |       ...      |
|       121       |        4        |      4.234      |        3       |
|       122       |        4        |      4.234      |        3       |
+-------------------------------------------------------------------+
```

Figure 10: Output of Listing 7.

```
1 from clusterx.correlations import CorrelationsCalculator
2 corrcal = CorrelationsCalculator("trigonometric", p_lat, cpool)
3
4 # Compute full correlations matrix for sset
5 x = corrcal.get_correlation_matrix(sset)
6
7 # Extract ab initio property values
8 p = sset.get_property_values("e_ads")
```

Listing 8: Code to set up the input matrix $\boldsymbol{X}$ (variable `x` in line 5) and the vector of calculated properties $\boldsymbol{P}$ (variable `p` in line 8) of Eq. (7).

```
1 from clusterx.estimators.estimator_factory import EstimatorFactory
2 from clusterx.model import Model
3
4 # Build and fit a ridge-regression estimator of scikit-learn
5 re = EstimatorFactory.create("skl_Ridge", alpha=1e-8)
6 re.fit(x,p)
7 ce_model = Model(corrcal, "e_ads", estimator = re) # Create a CE model
8 ce_model.report_errors(sset)
```

Listing 9: Creation of ridge-regression model with the `EstimatorFactory` and `Model` objects of `CELL`.

(line 7), and the errors are reported (line 8). The output, presented in Fig. 11, shows that the root mean square error (`RMSE`), the mean absolute error (`MAE`), and the maximum absolute error (`MaxAE`) of the fit are all zero. This is so because the number of clusters (122) is larger than the number of structures (50), thus, a perfect fit is possible. Conversely, the corresponding cross validation (CV) scores are different from zero. They express the ability of the built CE model to predict the properties of data not included in the fit [29]. The reason for a high CV score is either underfitting, or, what is the case now, overfitting. This can be avoided by searching on all possible subsets of clusters until finding the one for which the CV score is minimal. Such cluster-selection procedure, equivalent to regularizing with the ℓ_0-norm (see Sec. 2.3), is very demanding, since the number of subsets increases exponentially, making the problem numerically tractable only for small clusters pools. Nonetheless, very good approximations to the optimal solution exist.

Both the ℓ_0 solution as well as approximations to it are available through the `ClustersSelector` class of `CELL`. Although this class can be used independently, CE models can be easily built through a helper class called `ModelBuilder`. It encapsulates both the cluster selection and the estimator construction. In this example, we use the `ModelBuilder` class to find optimized models using two strategies. In the first one, the optimal solution is searched among sets of clusters of increasing radius and increasing number of points: For a given cluster to be present in a set, all other clusters with smaller radii and smaller numbers of points must be present as well. The other strategy makes use of the least absolute shrinkage operator (LASSO), which is a good approximation to the ℓ_0 solution under certain conditions (see Sec. 2.3).

In the following listing, the code for an optimization with the first strategy is shown:

```
+-------------------------------------------------------+
|                Report of Fit and CV scores            |
+-------------------------------------------------------+
|                   |        Fit        |        CV     |
+-------------------------------------------------------+
|       RMSE        |      0.00000      |      0.01412  |
|        MAE        |      0.00000      |      0.01146  |
|       MaxAE       |      0.00000      |      0.03907  |
+-------------------------------------------------------+
```

Figure 11: Output of Listing 9.

```
+-------------------------------------------------------+
|                Report of Fit and CV scores            |
+-------------------------------------------------------+
|                   |        Fit        |        CV     |
+-------------------------------------------------------+
|       RMSE        |      0.00469      |      0.00677  |
|        MAE        |      0.00363      |      0.00505  |
|       MaxAE       |      0.01378      |      0.02306  |
+-------------------------------------------------------+
```

Figure 12: Output of Listing 10.

```
1 from clusterx.model import ModelBuilder
2 mb = ModelBuilder(
3    selector_type="subsets_cv",
4    selector_opts={"clusters_sets":"size"},
5    estimator_type="skl_Ridge",
6    estimator_opts={"alpha":1e-8, "fit_intercept":True})
7
8 ce_model = mb.build(sset, cpool, "e_ads")
9 ce_model.report_errors(sset)
```

Listing 10: Creation of a CE model with the helper class `ModelBuilder`.

Here, the `ModelBuilder` class is imported (line 1) and instantiated in lines 2-6. Its initialization requires to specify (i) what strategy to employ to select the optimal model, and (ii) what estimator to use once the optimal set of clusters is determined. For (i), the arguments `selector_type` and `selector_opts` are set to `"subsets_cv"` and `"clusters_sets":"size"`, respectively, indicating that the optimal solution will be searched among sets of clusters of increasing size, as explained above. For (ii), the values assigned to the arguments `estimator_type` and `estimator_opts` have the same meaning as in Listing 9. The keyword `"fit_intercept"` is set to `True` (line 6), which amounts to add the empty cluster in the expansion (see Sec. 2.2). Using this setup, the CE model is built in line 8, and the errors are reported in line 9. The output is shown in Fig. 12. As compared to the previous CE model (Fig. 11), here the fitting errors are not zero, however the generalization error (see column `CV`) is smaller, indicating that this model yields better predictions on new configurations, *i.e.*, configurations not contained in the training set `sset` used to build the model. Figure 13 shows the predicted energies, both for the fit (black points) and CV (red points) as a function of Pt concentration. The figure is created as in Listing 6, by adding the argument `(..., cemodel = ce_model, ...)` to the function call in line 2.

It is interesting to consider in more detail the cluster optimization performed by the `ModelBuilder` class in Listing 10. Figure 14 shows the RMSEs for both fit and CV for all sets of clusters considered, with the respective cardinality given in the abscissa. The optimal set, indicated with a red circle, contains 12 clusters.

The LASSO selector can be used by replacing lines 3 and 4 in Listing 10 with

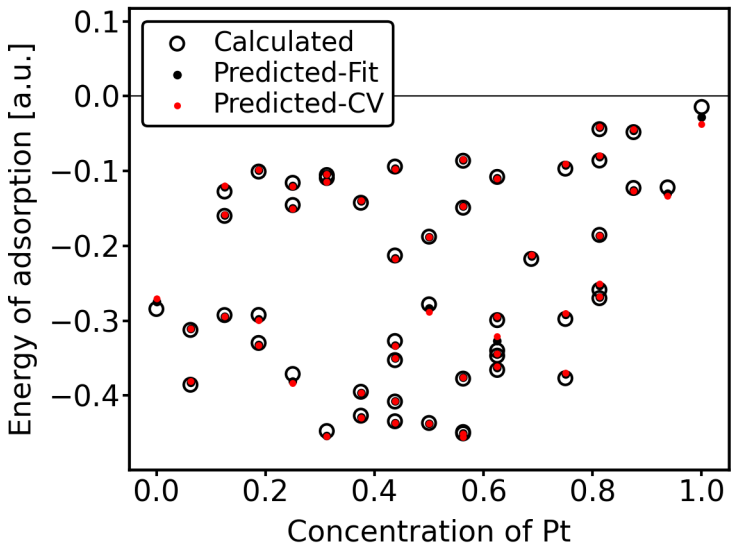

Figure 13: Predicition of *ab initio* adsorption energies and CVs, using the `ModelBuilder` class of `CELL` (Listing 10).

	Fit	CV	#Clusters
No optimization	0.00000	0.01412	122
Subset-CV	0.00469	0.00677	12
LASSO-CV	0.00095	0.00183	22

Table 1: Errors (RMSE) for models obtained with different strategies.

```
selector_type="lasso_cv",
selector_opts={'sparsity_max': 1e-2,'sparsity_min': 1e-6},
```

Listing 11: Creation of a CE model with LASSO using the `ModelBuilder` class.

In this case, the size of the cluster sets is controlled by the sparsity parameter λ in the ℓ_1 penalization term $\phi = \lambda\|\boldsymbol{\mathcal{J}}\|_1$ (see Sec. 2.3): Larger values of λ yield sparser models with smaller numbers of clusters and, conversely, smaller values of λ yield larger cluster pools and eventually overfitted models. The interplay between sparsity and predictive power is shown in Fig. 15. As expected, the RMSE of the fit decreases monotonously for decreasing sparsity, *i.e.*, the larger the cluster pool, the better the fit. Table 1 shows the values of the RMSE errors for the three employed strategies for model construction.

4 Thermodynamics

The stability of multi-component alloys at finite temperatures is determined by the free energy, $F = U - TS$, where U is the internal energy and S the entropy. S consists of at least three terms, namely, electronic, vibrational, and configurational contributions. The latter is usually approximated with the simple formula $S = -k_B \sum_i n_i x_i \log(x_i)$, with n_i and x_i being, respectively, the

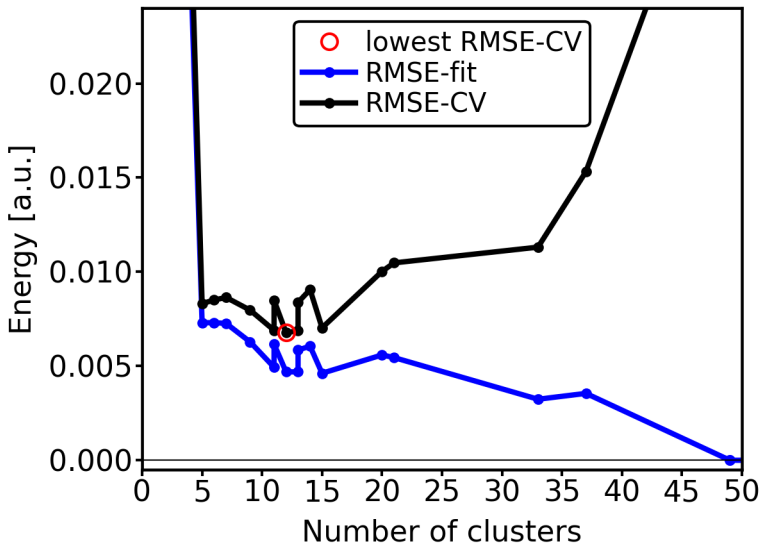

Figure 14: Optimization performed by the clusters selector.

number of sites and fractional concentration of sublattice i, and k_B the Boltzmann constant. Although being practical, it neglects interactions between substituent species, which can be crucial in determining the properties of complex alloys [30]. In `CELL`, instead, the configurational entropy, including the effects of interactions, is accurately accounted for in the calculation of thermodynamic properties. This is achieved by employing different sampling methods, including the Metropolis Monte Carlo sampling and the Wang-Landau sampling, together with a CE model for the internal energy of the alloy. In this section, the application of these methods is demonstrated on the Pt/Cu(111) surface alloy using the CE model of Sec. 3.

As input, the sampling procedures require a CE model that predicts the energy of newly proposed structures during the sampling. The simulation cell and the thermodynamic ensemble must also be specified. In the canonical ensemble, for instance, it is necessary to provide the substituents' concentrations in the sublattices. A detailed documentation is provided in Ref. [4]. Listing 12 shows the initialization of the `Model` object in line 3, corresponding to the adsorption energy, E_{ads}, for the binary surface alloy (Eq. (8) with $n_O = 0$). The simulation cell is created in line 7 by instantiating the `SuperCell` class that takes as arguments the parent lattice and a transformation matrix (`[[4,0],[-2,4]]`), which defines a rectangular simulation cell with 16 substitutional surface sites. In the examples demonstrated below, we perform samplings in the canonical ensemble, *i.e.*, with a fixed Pt concentration as specified in line 10. (Note that the dictionary key with value `1` refers to the sub-lattice corresponding to the top-most atomic layer of the Cu(111) surface.) Having 4 Pt atoms in 16 sites, leads to the stoichiometry of Cu_3Pt at the surface.

With the CE model, the simulation cell, and the concentration already set up, we can start a canonical sampling. The Metropolis Monte-Carlo method is explained in Sec. 4.1, and the Wang-Landau method in Sec. 4.2.

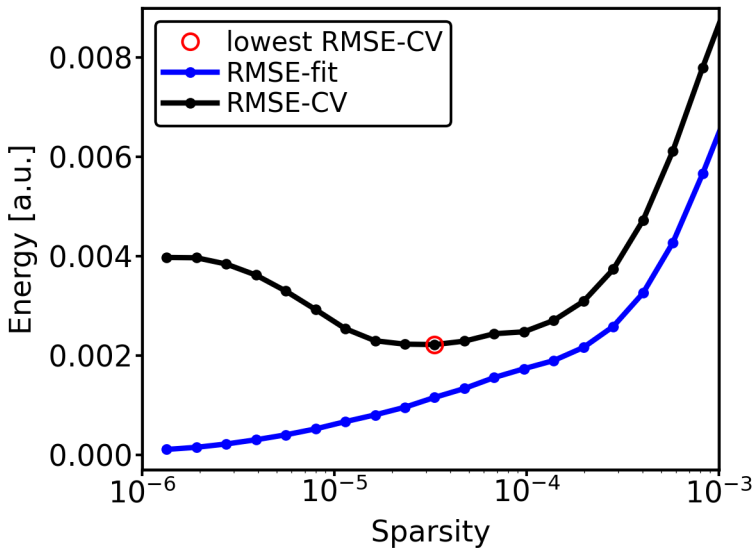

Figure 15: Optimization performed by the `ClustersSelector` using LASSO.

```
# Read the cluster expansion model for the absorption energy
from clusterx.model import Model
ce_model_ptcu = Model(filepath = "ce_model_ptcu.json")

# Create simulation cell
from clusterx.super_cell import SuperCell
scell_ptcu = SuperCell(ce_model_ptcu.get_parent_lattice(),[[4,0],[-2,4]])

# Define concentration for the canonical sampling
nsubs={1:[4]}
```

Listing 12: Initialization of a sampling procedure.

4.1 Metropolis Monte Carlo sampling

A widely used method to study finite-temperature properties in alloys is the Metropolis Monte-Carlo method [31, 32]. In this sampling method, trial moves made by swapping two atoms from randomly chosen crystal sites, are accepted with the probability

$$P(E_0 \to E_1) = \min\left[\exp\left(-\frac{E_1 - E_0}{k_\mathrm{B}T}\right), 1\right], \tag{9}$$

where E_0 is the energy of the initial structure, E_1 the energy after swapping the atoms, and $\exp(-E/k_\mathrm{B}T)$ the Boltzmann probability distribution at temperature T.

Listing 13 demonstrates how to perform a Metropolis Monte-Carlo (MC) sampling with `CELL` and how to compute the specific heat at a given temperature after the sampling. First, the number of sampling steps `nmc` is specified in line 2. A set of temperatures, `temps`, is given in line 5. Here, the units of the temperature and k_B have to be consistent with the energy units from the CE model, which is eV per substitutional site in our case. Thus, it is convenient to

```
# Define number of sampling steps
nmc=100000

# Define list of temperatures for the sampling (in Kelvin)
temps = range(1500,0,-50)

# Load Boltzmann constant in eV / K from ASE
from ase.units import kB

# Get number of Cu-Pt sites
nsites = scell_ptcu.get_index()

# Perform Metropolis Monte Carlo samplings
from clusterx.thermodynamics.monte_carlo import MonteCarlo
mc = MonteCarlo(ce_model_ptcu, scell_ptcu, ensemble = "canonical", nsubs=nsubs)

cp_mc = []
for temp in temps:
   traj = mc.metropolis(nmc, [kB,temp,1/nsites], write_to_db = True)
   cp = traj.get_average_value("Cp", equilibration_steps = int(nmc/5))
   cp_mc.append(cp)
```

Listing 13: Metropolis Monte-Carlo sampling with `CELL`.

use eV/K for k_B, and K for temperature.

The MC sampling requires a `MonteCarlo` object, which is created in line 15 by instantiating the `MonteCarlo` class of `CELL`. For the initialization, we indicate the CE model (`ce_model_ptcu`), the simulation cell (`scell_ptcu`), the ensemble type (`"canonical"`), and the concentration (`nsubs`, see Listing 12). Using this instance (`mc`), an MC sampling is executed in line 19 by calling the method `metropolis` of `mc` for each temperature `temp` in the list `temps`. The number of sampling steps, `nmc`, is passed as argument. The product Π of the elements in the second argument, `[kb,temp,1/nsites]`, enters the exponent of the acceptance probability as $\exp(-\Delta\hat{E}/\Pi)$, with $\Delta\hat{E}$ being the energy change predicted by the CE model. Since our CE model predicts energies per site, and the Boltzmann factor must be computed with total-energy changes, we include the factor `1/nsites` in the list. After successful completion of the sampling, a `MonteCarloTrajectory` object of `CELL` is returned and assigned to the variable `traj`. This object contains detailed information of the MC sampling trajectory, as *e.g.*, the energies of the visited structures and their configuration. Thus, any information of the MC trajectory can be read after the sampling procedure. For instance, by calling the method `get_average_value` of the `traj` object in line 20, we compute for each sampled temperature, the specific heat C_p, by passing the argument `"Cp"`. The result, shown in the top panel of Fig. 16 (dark red dots connected by dashed lines) exhibits a maximum at $\sim 450\,\mathrm{K}$.

We can repeat the MC simulation with a larger simulation cell, *e.g.*, by passing the transforma-

tion matrix `[[8,0],[-4,8]]` in the initialization of the `SuperCell` object (line 7 of Listing 12), which produces a rectangular simulation cell of 64 sites. The resulting specific heat is shown in the top panel of Fig. 16 (light red dots connected by dashed lines). As compared to the previous result with a smaller simulation cell, the peak becomes higher and narrower, and shifts to a lower temperature of around 350 K. This behavior signals an order-disorder phase transition. Indeed, from the MC trajectories, we see that the dominant configuration at low temperatures is the ground-state structure depicted in the left panel of Fig. 17. It reveals a p(2x2) ordering of the Pt atoms, as expected from Ref. [22]. For comparison, on the right side of Fig. 17, a snapshot of the trajectory at 1500 K is depicted, which looks rather disordered.

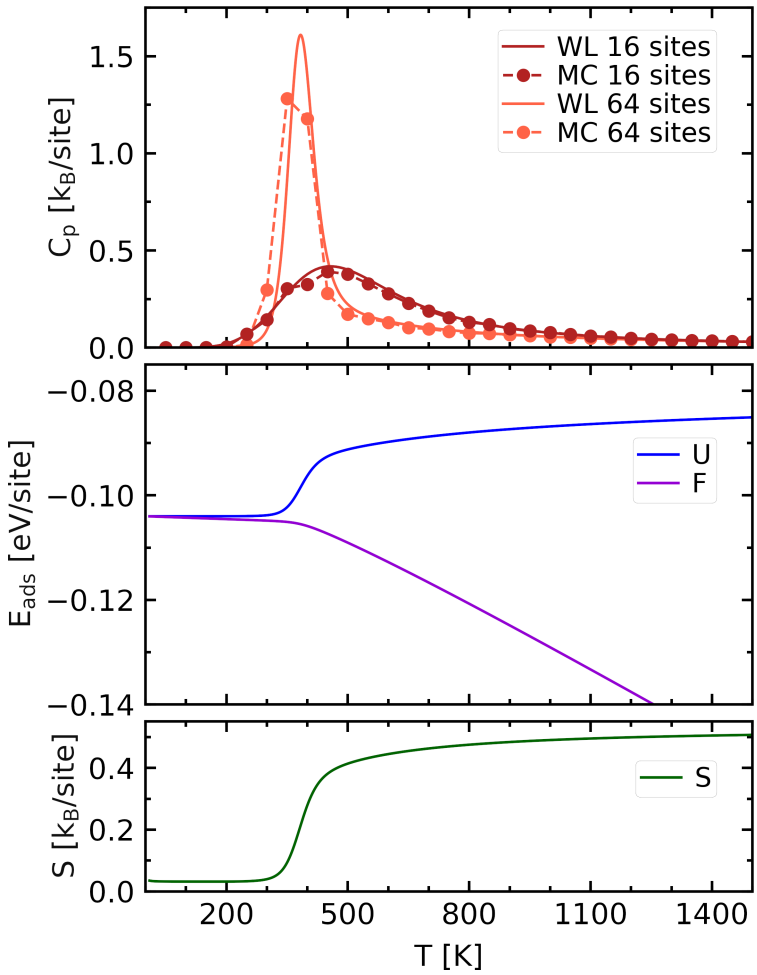

Figure 16: Thermodynamic properties of the Pt/Cu(111) surface alloy as a function of temperature. Top: Specific heat for simulation cells with 16 (dark red) and 64 surface sites (light red), respectively, obtained with MC sampling (dashed lines) and the WL method (solid lines). The middle panel shows the internal energy, U, and the free energy, F, for the simulation cell with 64 sites obtained by the WL method; the bottom panel the respective entropy, S.

4.2 Wang-Landau sampling

The Wang-Landau (WL) method [33, 34] aims at reducing the simulation effort by obtaining the configurational density of states, $g(E)$ –a temperature-independent quantity– directly within the sampling procedure. Once $g(E)$ is known, the thermodynamic properties can be easily obtained

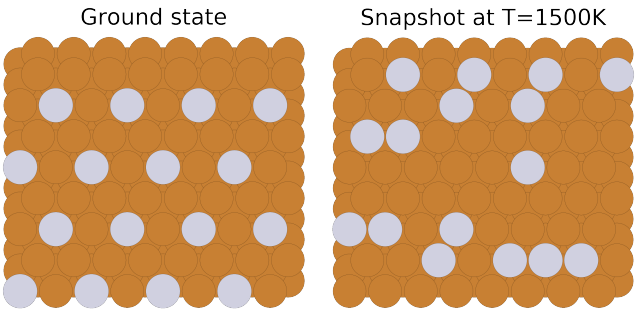

Figure 17: Ground-state structure (left) and snapshot at $T = 1500\,\mathrm{K}$ (right) of the Pt/Cu(111) surface alloy in a simulation cell with 64 surface sites. Brown circles represent Cu atoms, gray circles Pt atoms.

at any temperature. This represents an enormous advantage as compared to MC simulations, particularly for quantities that require thermodynamic integration for its evaluation, like the free energy [35, 36]. In the WL approach, the latter is readily available, since the partition function Z is directly obtained from $\int dE g(E) \exp(-E/k_B T)$. The WL algorithm is described in detail in Refs. [33, 34], and applications of this method can be found in Refs. [12, 37, 38]. In the following, its workflow is briefly explained, and its implementation in `CELL` is exemplified by the Pt/Cu(111) surface alloy.

In the WL sampling, the energy space is sampled with the probability proportional to $1/g(E)$ which, for the exact $g(E)$, should produce a flat histogram $H(E)$ of the visited energies. At the start, however, $g(E)$ is unknown. Thus, the energy space is discretized into energy bins E_i, where each bin is assigned *a priori* a uniform density of states $g(E_i) = 1$. A newly proposed structure with energy E_{new} and density of states $g(E_{\mathrm{new}})$ (equal to 1 at the start) is accepted with the probability

$$P(E_{\mathrm{old}} \rightarrow E_{\mathrm{new}}) = \min\left[\frac{g(E_{\mathrm{old}})}{g(E_{\mathrm{new}})}, 1\right], \tag{10}$$

where E_{old} and $g(E_{\mathrm{old}})$ are the energy and density of states of the initial structure, respectively. If the trial structure is accepted (rejected), both $g(E)$ and $H(E)$ for the energy bin containing E_{new} (E_{old}), are updated according to the following rule: $g(E) \rightarrow f\, g(E)$ and $H(E) \rightarrow H(E)+1$. With the multiplication factor f kept fixed, the sampling is continued until the histogram satisfies a predetermined flatness condition. Then, the modification factor f is reduced and the histogram restarted. The complete sampling procedure consists of a nested loop, with the inner loop generating the flat histogram and the outer loop reducing the modification factor. The accuracy of the final $g(E)$ is determined by the final flatness condition and modification factor.

The WL algorithm is implemented within `CELL` in the `WangLandau` object and can be used as shown in Listing 14. A `WangLandau` object is initialized in line 2 with the CE model of E_{ads}, the simulation cell, and the given concentration, similar to the initialization of the `MonteCarlo` object in Listing 13. The WL sampling is then executed in line 4 by the method `wang_landau_sampling`. Here, `energy_range` and `energy_bin_width` define the energy window and bin width of the histogram, respectively. The modification factor f and the flatness condition are predefined by default values that can be changed by the user. The initial modification factor $f = e$, is reduced by $f \rightarrow \sqrt{f}$ in each iteration of the outer loop, until the final modification factor is

```
1 from clusterx.thermodynamics.wang_landau import WangLandau
2 wl = WangLandau(ce_model_ptcu, scell_ptcu, nsubs)
3
4 cdos = wl.wang_landau_sampling(energy_range=[-0.103,0.0], energy_bin_width=0.002,
      f_range=[math.exp(1), math.exp(1e-3)])
5
6 # Temperature range for the calculation of the specific heat
7 temps = range(1500,0,-1)
8 cp = cdos.calculate_thermodynamic_property(temps,"Cp")
```

Listing 14: Wang Landau sampling with `CELL`.

$f = \exp(10^{-3})$ (see argument `f_range` in line 4). The condition for determining the flatness of the histogram is $\min[H(E)] > c\,\overline{H(E)}$, where the overline indicates the mean value, and c is a real number. Starting with $c = 0.5$ in the first iterations, it is then increased once every few iterations of the outer loop, until reaching $c = 0.9$. More details and ways to control the default behavior can be found in Ref. [4].

The algorithm returns a `ConfigurationalDensityOfStates` object, assigned to the variable `cdos` in line 4 of Listing 14. Using this object, several thermodynamic quantities can be computed directly for an arbitrary number of temperatures. For instance, line 8 tells to compute the specific heat for the temperatures given in line 7. The result is shown in the top panel of Fig. 16 (dark red solid line), together with that of the larger simulation cell (light red solid line) in comparison with the MC simulations. Since the evaluation of C_p from $g(E)$ is computationally less demanding than the Metropolis Monte-Carlo method presented in the previous section, we can determine the transition temperature more accurately. In the lower panels of Fig. 16, the internal energy, U, the free energy, F, and the entropy, S, are presented. The entropy clearly increases with increasing temperature, indicative of an order-disorder phase transition. Note that the results presented here, can only show trends, but are not fully quantitative, since the CE model of the absorption energies is trained with model energies (see Sec. 3.)

5 Applications

In this section, we demonstrate the application of `CELL` to the binary alloy Si-Ge and to the clathrate $Ba_8Al_xSi_{46-x}$ as an example of a complex intermetallic alloy.

5.1 Si-Ge alloy

The binary alloy Si_xGe_{1-x} exhibits a miscibility gap, without forming ordered structures [39, 40], *i.e.*, it has a strong tendency to separate into almost pure Si and Ge phases. This tendency decreases with increasing temperature until a critical temperature is reached beyond which the fully disordered phase is stable at any Si concentration. The theoretical description of this demixing transition has typically been done using Metropolis Monte Carlo simulations in the

grand canonical ensemble [6, 40–43]. Here, we approach it using the Wang-Landau method in the canonical ensemble. This approach is interesting because it allows access to the phase-separation region, which is inaccessible in the grand canonical ensemble. We also describe the bowing of the lattice constant at different concentrations and temperatures. These results are compared with available experimental data.

Using a 16-atom supercell with lattice vectors $(a, a, 0)$, $(a, 0, a)$, and $(0, a, a)$, a being the lattice constant of the parent diamond crystal, we generate a set of 43 structures with random configurations containing $n_{\mathrm{Ge}} = 0, 1, 2, ..., 16$ Ge substituents. The lattice constants are optimized, and the atomic positions relaxed until the forces are smaller than 5×10^{-3} eV/Å. The *ab initio* energies are calculated by density-functional theory (DFT) [44, 45] with the all-electron, full-potential electronic-structure code FHI-aims [46, 47]. Exchange and correlation effects are treated within the generalized gradient approximation, employing PBEsol [48]. The basis set is determined by "tight" settings. A $10 \times 10 \times 10$ $\boldsymbol{k}$-point grid is used for integrations in the supercell Brillouin zone (BZ).

The data generated in this way serve as the training set for CE models of the energy of mixing per atom, E_{mix}, and of the lattice parameter, a_0. The former is defined by $E_{\mathrm{mix}}(x) = E(x) - [(1-x)E_{\mathrm{Si}} + xE_{\mathrm{Ge}}]$ with $E(x)$ being the total energy per atom of the compound with Ge concentration x, and E_{Si} and E_{Ge} the energies of the pristine Si and Ge solids, respectively. The CE models are built using a pool of clusters as shown in the right panel of Fig. 18, containing all clusters up to three points contained in the supercell: Besides the empty and a single one-point cluster, there are four 2-point clusters and six 3-point clusters. The cluster selection is performed by a combinatorial search on all possible subsets of clusters, with the condition that the first three (*i.e.*, the empty, the one-point, and the first-neighbor 2-point cluster) are included. The learning curves for the two CE models are shown in the left panels of Fig. 18. Here, the model which minimizes the RMSE-CV, marked by the red diamond, is selected. Every dot (plus sign) indicates the RMSE-fit (RMSE-CV) of a single trial model. The optimal model for E_{mix} contains 8 clusters (blue background in the right panel), while for a_0 it contains 11 clusters. The solid orange line joins the models with optimal RMSE-CV as a function of the number of clusters, and the green solid line indicates the corresponding RMSE-fit for these models. While the latter decreases monotonously, the former may have a minimum as in the case of a for 11 clusters, signaling overfitting for models with more clusters. The ECIs of the final CE model for E_{mix} are displayed in the middle panel of Fig. 18. The cluster radii are given in units of the nearest-neighbor distance $\mathrm{R_{nn}}$, *i.e.*, the shortest distance between two Si atoms in the parent lattice. Most ECIs are negative, in accordance with the well known tendency of SiGe to phase-separate at low temperatures.

The accuracy of these models becomes more clear in Fig. 19 where excellent agreement between the DFT data (red circles) and the CE predictions (black dots) is observed. With these models, property predictions are made for all possible derivative structures of up to 16 atoms. The generation of the derivative structures is done with `CELL` using the algorithm of Ref. [49]. The resulting predictions are shown with blue dots. The fact that for all structures $E_{\mathrm{mix}} \geq 0$ means that at zero temperature, the system would favor a demixed state at all concentrations, without forming ordered structures.

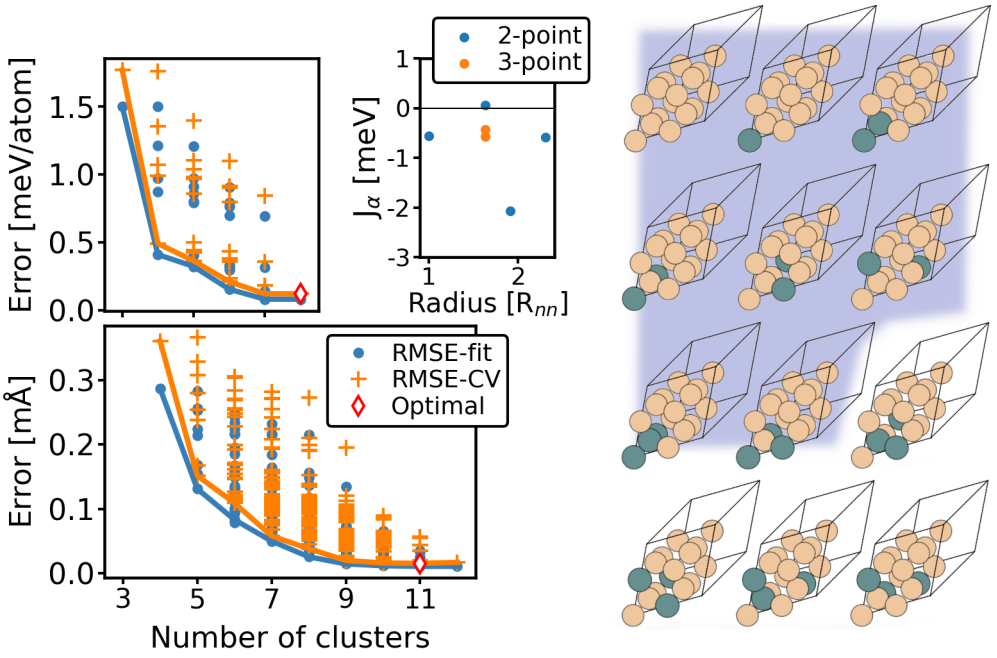

Figure 18: CE models for SiGe. Upper left panel: Optimization of the CE model for the energy of mixing. Middle panel: Effective cluster interactions for the corresponding optimal CE model. Lower left panel: Optimization of the CE model for the lattice parameter. Right panel: Pool of clusters for building the CE models for the lattice constant and for the energy of mixing (blue background).

A negative bowing of the lattice constant is observed in the lower panel of Fig. 19, which indicates the departure from Vegard's law, which reads $\Delta a(x) = a(x) - [(1-x)a_{\mathrm{Si}} + x\, a_{\mathrm{Ge}}]$, with $a(x)$ being the lattice constant of a structure with Ge concentration x and a_{Si} and a_{Ge} those of the pristine solids. For the perfectly random alloy, the cluster correlations can be computed analytically with `CELL`, so that the CE model can also be employed to compute the lattice parameter in this limiting case. This is shown by the solid red line, which yields Δa values very close to the random structures used for training. The qualitative behavior is also similar to the experimental data, shown in the figure by red dots with error bars. Still, the experimental values are visibly smaller in magnitude than the prediction for the random alloy. This difference could in part be due to temperature-dependent configurational effects, as will be discussed below.

Using these models, we study the temperature dependence of the demixing transition at 50% Ge concentration for a cubic SiGe supercell containing 2744 atoms. Figure 20 shows the results of a Wang-Landau sampling performed in parallel using 32 CPUs. Each process performs sampling in different overlapping energy ranges, as indicated by different colors in the first three panels of the figure. The first panel shows the converged WL histograms per process converged up to a flatness condition of $c = 0.995$. Each histogram contains 22 bins with a bin width of 51 meV. In the last WL iteration, about $2 \cdot 10^8$ structures per energy range are visited. The normalized histograms, indicated by black dots, merge into a global flat histogram for the entire energy range considered. The latter is determined at the start of the simulation by considering the predicted energies of a fully demixed and a fully disordered structure, giving the minimum and maximum energies in the energy range, respectively.

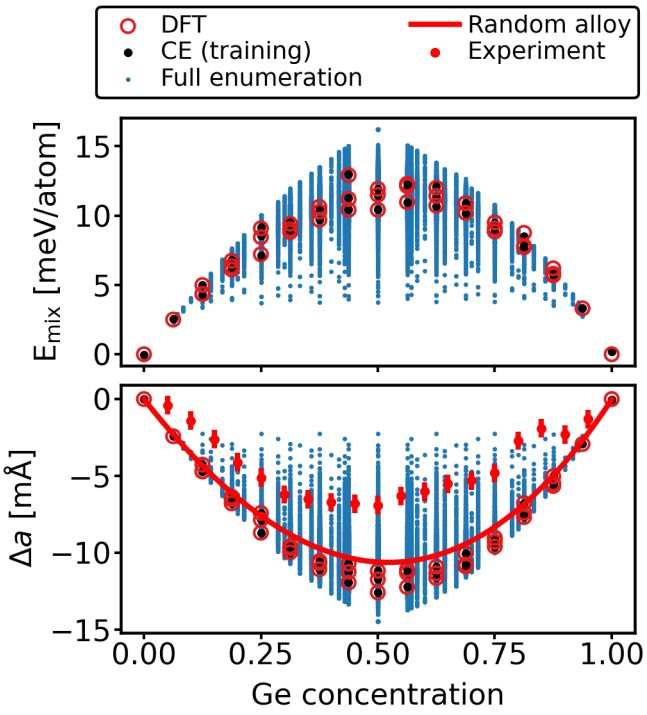

Figure 19: DFT training data (red circles), CE predictions after training (black dots), and full structure enumerations (blue dots) for the CE of the energy of mixing (upper panel) and the lattice parameter (bottom panel) for the SiGe alloy as a function of Ge concentration. For the lattice constant, the predictions of the CE model for the perfectly random alloy are shown by a red solid line, experimental values, taken from Ref. [50], by red dots.

The logarithm of the configurational density of states, $log(g)$, is evaluated for all processes and converged up to a final modification factor f satisfying $ln(f) = 2^{-21}$. Upon reaching convergence, each process i yields a configurational density of states $C_i\, g(E)$, with an arbitrary normalization C_i. This means that the $log(g)$ of contiguous processes i and $i+1$ are shifted by an additive constant $log(C_i/C_{i+1})$. Therefore, to extract $g(E)$ with a common normalization factor, one has to shift the parts of $log(g)$ by appropriate additive constants. As described in Ref. [51], these are determined by finding the energies at which the microcanonical temperature $T_{mc} = (k_B \mathrm{d}ln(g)/\mathrm{d}E)^{-1}$ from adjacent energy ranges overlap best. The third panel of Fig. 20 shows T_{mc} for each process. Here, the red arrow in the inset illustrates the point of best overlap of T_{mc}'s for two contiguous ranges. The resulting $log(g(E))$ is shown by the black line in the second panel. It is normalized such that the lowest-energy configuration has a degeneracy of one. It is to be noted though, that the results presented below do not depend on the chosen normalization.

From $log(g)$, one can easily evaluate the canonical probability distribution $P(E,T)$ at any temperature. This is useful for computing thermodynamic averages of different quantities, as will be shown below. In the fourth panel of Fig. 20, $log(P) - log(P_{max})$ is shown for various temperatures. As expected, the energy at which P is maximum increases monotonically with temperature. At large temperatures, the maxima approach the limit of the fully random structure. This is why the maxima for $T = 300K$ and $T = 350K$, which are above the demixing transition temperature, are close to each other. Since, importantly, $log(P)$ always shows a single maxi-

mum, in agreement with Ref. [41], the transition does not seem to be of first order, at least for the supercell sizes considered here.

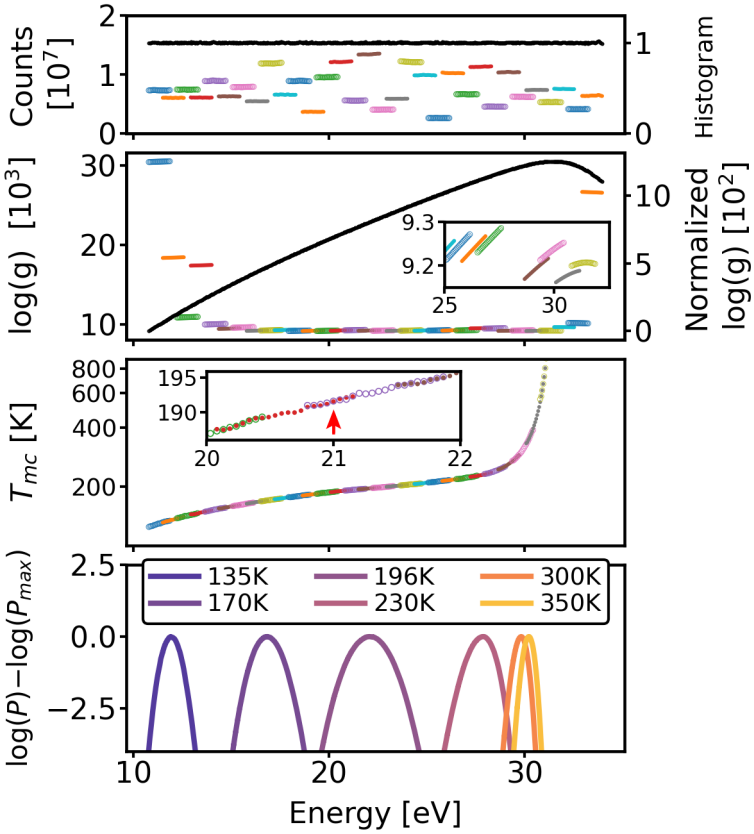

Figure 20: Parallel Wang-Landau (WL) sampling with `CELL` for a supercell of 2744 atoms, using 32 processors. Top panel: WL histograms per CPU (colored dots and circles, left y-axis) and normalized histogram (black dots, right y-axis). Second panel: logarithm of the configurational density of states, $\log(g)$, per CPU (colored dots and circles, left y-axis), and the normalized $\log(g)$ (black dots, right y-axis). The inset shows a close up to highlight the energy dependency of the $\log(g)$ in each process. Third panel: microcanonical temperature, T_{mc}, per processor. The arrow in the inset highlights the point of best overlap of T_{mc}'s. Bottom panel: logarithm of the canonical probability distribution, $\log(P) = \log(g) - \beta E$, for selected temperatures. All quantities are shown as a function of energy.

The demixing transition is more closely analyzed in Fig. 21. Here, the specific heat at constant pressure, $C_p(T) = (\langle E^2\rangle - \langle E\rangle^2)/k_B T^2$, with $\langle E^n\rangle = \int dE\ E^n P(E,T)$, is computed using $P(E,T)$ from Fig. 20. The calculation is performed for three supercell sizes, containing 1000, 1728, and 2744 atoms, respectively. The specific heat peaks at a temperature that increases for increasing system size. For the largest system, the maximum is at about T=196K. At T$\sim$135K, C_p displays a shoulder and at T>196K it decays monotonously to zero. It is interesting to explore what kind of structures are present at different temperatures, which are indicated by thin vertical dashed lines. These are obtained by taking a sample from a microcanonical sampling at a narrow energy range centered at the corresponding maxima of $P(E,T)$ (see bottom panel of Fig. 20) and are shown in the right panel of Fig. 21. At about 135K, the system is essentially separated into a pure Si and a pure Ge phase, with very little dissolution of one species into the

bulk of the other. At 170K and 196K –the maximum of C_p– this dissolution increases, indicating the proximity of the mixed state. Just above the maximum, at T=230K and in the region of small C_p around T=300K, the structures appear fully mixed and increasingly disordered. These observations suggest a transition temperature of about T_c ∼200K. This value lies in between various values from previous simulations in the literature, such as 360K in Ref. [39], 170K in Refs. [40, 52], 320K in Ref. [41], 247K in Ref. [42], or 325K in Ref. [6].

For each temperature shown in the upper left panel of Fig. 21, $\Delta a(T)$ is estimated in a *single-shot* procedure. It consists of taking a single sample from a microcanonical sampling at an energy that maximizes $P(E,T)$ (as shown in the right panel of the figure) and then using the CE model for a to predict Δa. Δa decreases monotonously with temperature as shown in the lower panel of the figure (solid red line). At around the lowest computed temperature (T=135K), it approaches zero, which is the value predicted for the perfectly phase-separated SiGe alloy. Above T_c, at $T = 300K$ and $350K$, Δa is very close to the value predicted for the perfect disordered solid, as expected. At $T = 196K$, our result is close to the experimental value reported in Ref. [50]. These findings imply a negative expansion coefficient in the transition region, purely due to configurational effects. This region is typically inaccessible to experiments, which usually deal with the homogeneous disordered phase above the transition temperature. In this case, the configuration being in the disordered state, does not change with temperature and due to anharmonicities of the crystal, the measured expansion coefficient is positive and estimated to be about 5.4 $\times 10^{-6} K^{-1}$ at a concentration of 51.3 % Ge [50]. Note that our calculations neglect anharmonic effects. These would presumably reduce the magnitude of the predicted Δa for increasing temperatures.

5.2 Si-based clathrate compounds

We now demonstrate the application of `CELL` to the study of a complex alloy, the intermetallic clathrate $Ba_8Al_xSi_{46-x}$. Intermetallic clathrates belong to the class of inclusion compounds in which the host lattice forms cages that can enclose guest species. Their low thermal conductivity together with their highly tunable electronic properties make them ideal candidates for thermoelectric applications [53]. The unit cell of $Ba_8Al_xSi_{46-x}$ is shown in Fig. 22. It contains 54 atoms, 46 of which are tetrahedrally bonded and form the host lattice of Si and Al atoms (Wyckoff sites 24k, 16i, and 6c). The 8 Ba atoms occupy the cages in Wyckoff sites 2a and 6d.

The material's properties, such as electronic structure and energy of formation, depend strongly on both the Al concentration x and the crystal sites that they occupy, *i.e.*, the configuration [30, 38, 54]. Therefore, it is of paramount importance to find the configuration of the ground-state structures (GSS), in order to understand the material's physical properties. This is, however, a formidable task. Due to the incredible number of possible configurations, *i.e.*, to arrange the Al atoms in the unit cell (∼10^{11} for $x = 16$), approaches relying on a full enumeration of structures are infeasible. Therefore, in Ref. [30], some of us have devised a special iterative approach to find the GSS and build an accurate CE model. The workflow, implemented in `CELL`, is depicted in Fig. 23. It is particularly useful for being applied to alloys with complex parent lattices, where a full structural enumeration is out of reach. First, the *ab initio* energies of an initial set of random structures (left white box) are calculated (left yellow box), and an initial CE model

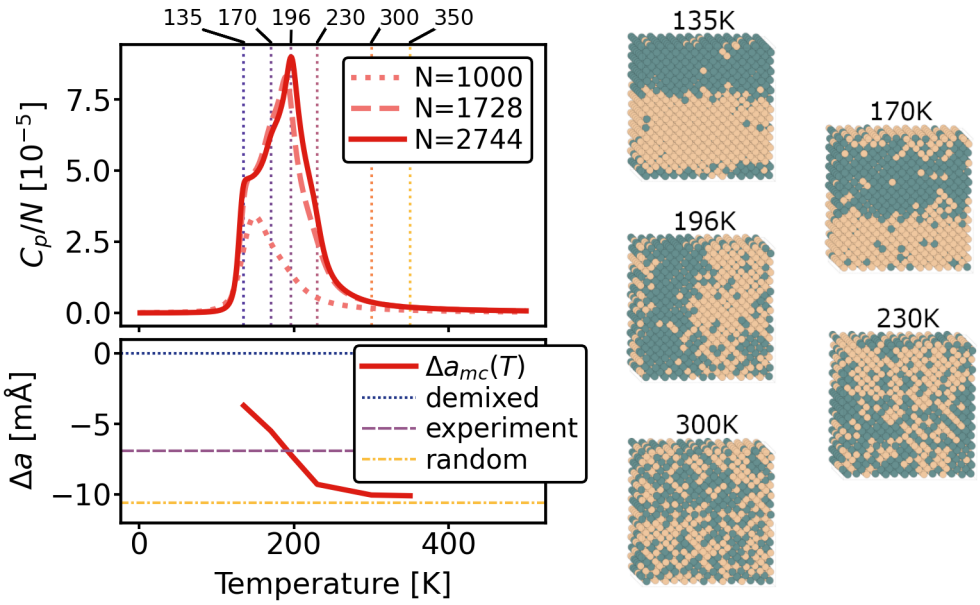

Figure 21: Upper left panel: Specific heat at constant pressure, C_p, as a function of temperature for different numbers of atoms N. The thin dashed vertical lines and corresponding labels on top indicate temperatures at which Δa is computed. Lower left panel: Δa as a function of temperature; the horizontal lines mark the predictions for Δa for the demixed state, the random state, and the experimental value from Ref. [50] at a Ge concentration of 51.3 %. Right panel: Samples from microcanonical sampling at energies that maximize $P(E,T)$ at the specified temperatures.

is built with this data set (left orange box). Next, a configurational sampling is performed (upper gray box); the predicted structures with the lowest energies that are non-degenerate to those already present in the data set (labeled "LND structures" in the white box on the right), are added to the data set; and an improved CE model is determined (right orange box). The performance of the CE model is then evaluated through, *e.g.*, cross validation. If the CV score is larger than the desired target precision, the CE model may be further improved by performing a new Metropolis sampling, as shown by the arrows. This is repeated until the CV score is smaller than the desired tolerance, and property predictions can be made.

Figure 24 shows the realization of this iterative cluster expansion for the clathrate alloy. The convergence of the CE model and the determination of the GSS are achieved in only 4 iterations. Here, the target accuracy of the CE model is 1meV/atom or less, since this allows one to correctly identify the GSS [30]. In the first iteration (top left), a set of 11 random structures with Al content in the range $x = 6-16$ is created, and the energy of mixing of each structure is computed *ab initio* with `exciting` using the functional PBEsol. `exciting` is a full-potential all-electron DFT package implementing the linearized augmented planewave + local-orbital method [55]. The energy of mixing per atom is defined by $E_{\text{mix}}(\sigma) = E(\sigma) - [\hat{E}_0(1-c) + \hat{E}_{46}c]$. Here, $E(\sigma)$ is the total energy per atom of the relaxed structure with atomic configuration σ, $\hat{E}_0$ ($\hat{E}_{46}$) is the predicted energy per atom of the hypothetical structure Ba_8Si_{46} (Ba_8Al_{46}), and $c = x/46$. Computational details for the calculation of $E(\sigma)$ are given in Ref. [30]. The *ab intio* computed energies are indicated by black circles in the figure. Following the workflow, an initial

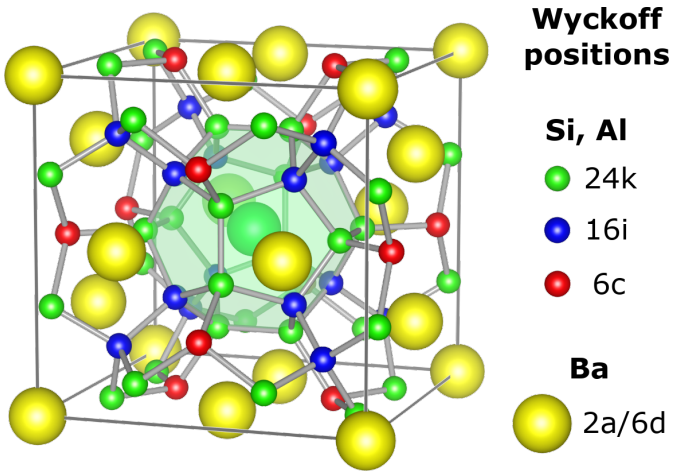

Figure 22: Primitive cell of the type-I clathrate $Ba_8Al_xSi_{46-x}$, consisting of 54 atoms. The Si-Al host lattice (Wyckoff positions 24k, 16i, and 6c) forms a cage-like structure that encloses 8 Ba guest atoms (Wyckoff positions 2a and 6d). One of the cages is highlighted.

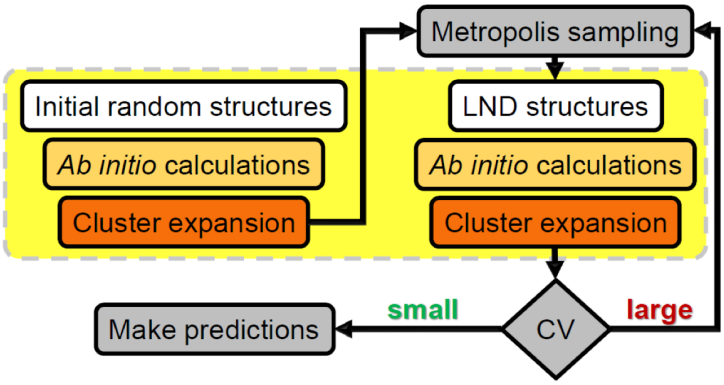

Figure 23: Workflow assembled with `CELL`'s modular structure, for an iterative construction of a CE model, based on structural sampling instead of full enumeration.

CE is constructed with these data. The predictions made with the corresponding CE model are represented by the black dots. Their RMSE-CV is 4.9 meV/atom. With this initial CE, a Metropolis sampling is performed for 1000K, with 5×10^5 steps per composition (gray dots) and the visited structures of lowest energy are identified (red dots). This step corresponds to the box "LND structures" of the workflow in Fig. 23. For these structures, *ab initio* calculations are carried out (red circles). For iteration 1, the disagreement between the predictions for the LNDs and their *ab initio* values for $x < 12$ is big. In iteration 2 (upper right panel), the newly computed data from the previous iteration are added to the training set, which now consists of 22 data points shown as black circles. A new CE model is fitted to these data. Its RMSE-CV of 4.4meV/atom is slightly smaller than that of the first iteration but still well above the desired accuracy of 1meV/atom. Thus, according to the workflow, a new Metropolis sampling is performed (gray dots), LND structures are identified (red dots), and *ab initio* calculations are performed for them (red circles). There is still a significant disagreement between predicted and *ab initio* energies for the new LND structures, which increases with increasing x. In iteration 3 (lower left panel) again the data from the previous iteration are added, so that the training

set (black circles) now consists of 33 structures. A new CE model trained with these data yields an RMSE-CV of 0.9 meV/atom, which is slightly below the desired accuracy threshold. Nonetheless, a new sampling is performed. The previously found GSS for $x < 16$ are confirmed, and three new LNDs are added for $x > 12$. For $x = 16$, a new ground-state structure is found. These four data points (red circles) are added to the training set in the fourth and final iteration (lower right panel); the resulting model has an RMSE-CV of 0.8 meV/atom. A new sampling confirms the previously found GSS, except for two new ones found for $x = 8$ and $x = 14$. The latter are quasi-degenerate to the previously found ones. The 3 new data points from iteration 4 are added to the training set and a final CE model is fitted.

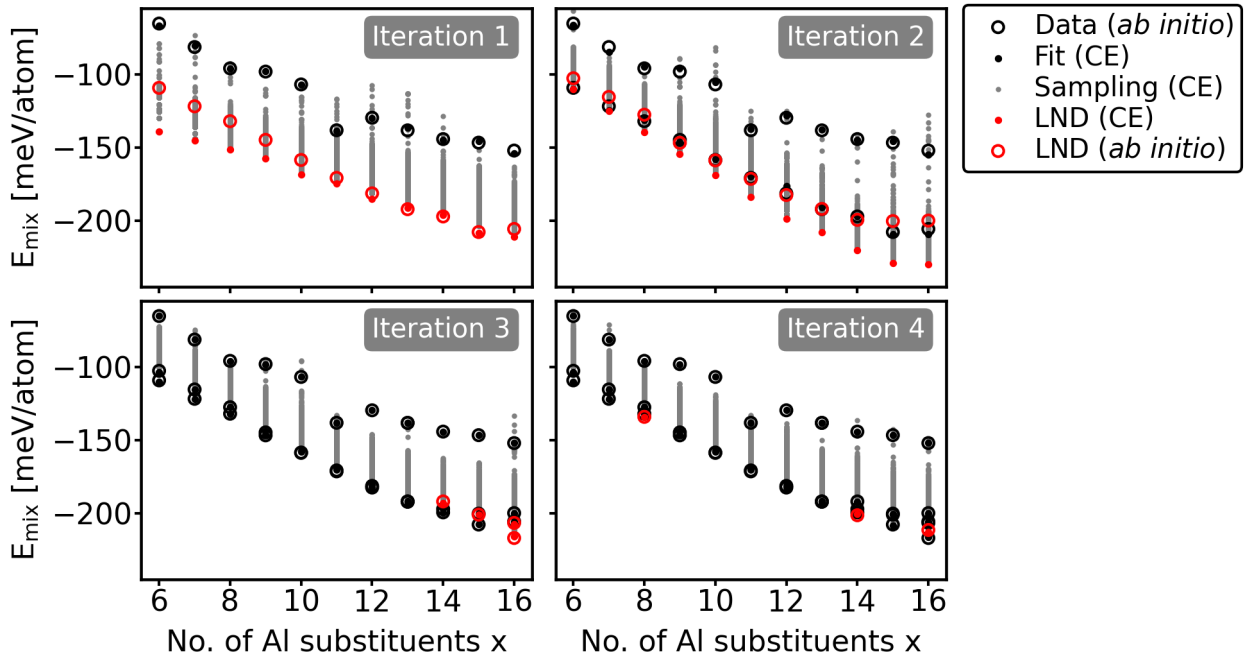

Figure 24: Construction of a CE model and ground state search for the clathrate alloy $Ba_8Al_xSi_{46-x}$, using the workflow of Fig. 23. Black circles: *ab initio* data used for training the CE model of the respective iteration. Black dots: CE predictions for the training data. Gray dots: Metropolis Monte Carlo samplings at 1000K with 5×10^5 steps per composition x. Red dots: CE predictions for the lowest non-degenerate (LND) structures found in each iteration. Red circles: *ab initio* results for the LND structures.

A closer look at the model performance for the different iterations is provided in Fig. 25. Focusing first on the fitting errors (left panel), the RMSE is above the accuracy threshold in the first two iterations and then decreases to a value of around 0.6meV/atom, below the desired accuracy. The median of the absolute errors follows a similar trend, but is considerably smaller in magnitude than the RMSE. The reason is that the latter is more sensitive to outliers. In the final model, a single outlier with an absolute error larger than 1meV/atom remains. For the generalization errors (right panel), a similar trend is observed. For the first two iterations, the RMSE-CV lies well above the accuracy threshold, but it stabilizes at a value of around 0.8meV/atom for the remaining models. There is good agreement between the RMSE-CV and the RMSE-Test, indicating that the estimates of the generalization error given by the RMSE-CV are good. The RMSE-Test is obtained from the red dots and red circles in each iteration shown in Fig. 24.

Thus, it represents the error on totally unseen data, not even used for CV.

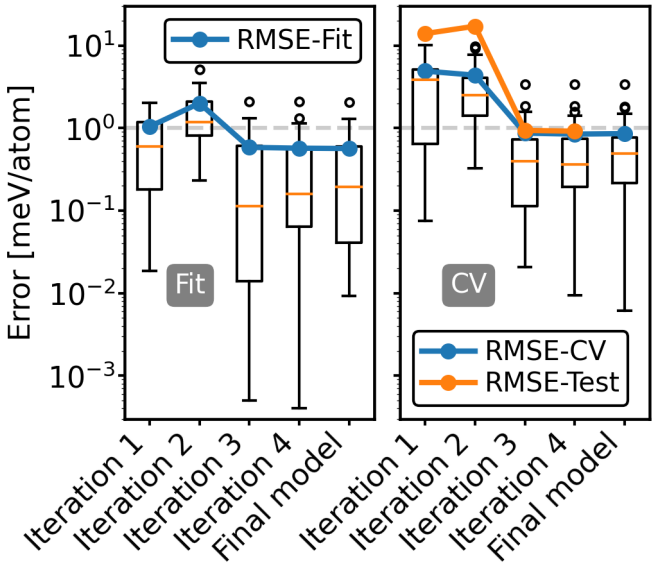

Figure 25: CE model performance corresponding to the iterations presented in Fig. 24. The box plots show the distribution of absolute errors, with the median indicated by thin orange lines and outliers by black circles. Left: fitting errors, right: generalization errors estimated with cross validation. RMSE-Test represents the errors from the red dots and circles of Fig. 24. The gray dashed horizontal line indicates the target precision.

The found CE model is able to make accurate predictions not only for GSS, but also for higher-energy structures. This allows a finite-temperature analysis using the thermodynamics modules of `CELL`. Such a study was performed in Ref.[38], where for the charge-balanced composition $x = 16$, a temperature-driven semiconductor-to-metal transition was found. The latter is accompanied by a partial order-disorder structural transition.

6 Conclusions

We have described in detail the Python package `CELL` for cluster expansion and for statistical thermodynamics. `CELL` provides a modular framework that allows for customized CE model building and integration into workflows. This paves the way to address a wide range of problems, as illustrated by various applications. The first one, has demonstrated the ability of `CELL` to create the CE model of a complex surface system, O-Pt/Cu(111). This system is characterized by the interplay of two binary sublattices, one describing Pt/Cu surface alloying and the other the oxygen surface adsorption. Such systems are of great interest, for instance, for applications in catalysis. The characterization of the structure at finite temperature revealed a temperature-driven order-disorder transition. The found ordered low-temperature phase is in agreement with experimental findings.

The second example showed the application of `CELL` to the binary semiconducting alloy Si-Ge. Using a combinatorial approach to model selection, accurate CE models for the mixing energy and lattice parameters were generated. The prediction of the energy of mixing for all derivative structures of up to 16 atoms confirmed the tendency of Si-Ge to separate into

pure Si and Ge phases. Consideration of the fully random alloy together with the CE model for the lattice constant, revealed a negative bowing of the lattice parameter, in agreement with experiments. We have characterized the demixing transition of Si-Ge in terms of the configurational density of states, the microcanonical temperature, the canonical probability distribution, the specific heat, and the thermal expansion. Our analysis, performed in the canonical ensemble, both contrasts with and nicely complements previous studies in the literature using the grand canonical ensemble. The latter yields homogeneous phases but prevents access to the phase-separated state, which is accessible in our study. In this example, we have showcased the parallel execution of `CELL`'s with supercells containing thousands of atoms. Inspection of structures from microcanonical sampling yielded a demixing transition temperature of around 200K, which is in the range of values reported in the literature.

The third and last example concerned the construction of a CE model for the energy of the complex clathrate alloy $Ba_8Al_xSi_{46-x}$. This material, with its 54 atoms in the unit cell, posed a formidable challenge to the tasks of building accurate CE models and finding the lowest-energy structures. We have addressed this problem by creating an iterative workflow that efficiently solves both problems simultaneously. The workflow uses continuously improved CE models to perform configurational sampling. These are used to identify low-energy structures that are iteratively added to the training set. With only four iterations, requiring the *ab initio* calculation of just 40 structures, all ground states in the range $x = 6-16$ are found. Analysis of the model's performance in terms of fitting and generalization errors revealed that convergence is already achieved by the third iteration.

In summary, `CELL` provides a comprehensive approach to cluster expansion, covering all aspects of model construction and thermodynamical analysis. `CELL` offers its users an efficient way to build CE models using machine-learning techniques and leveraging parallelization, seamlessly integrating with Python code and facilitating the interaction with *ab initio* packages.

7 Data and software availability

`CELL` is available at the Python Package Index (PyPI) repository `https://pypi.org/project/clusterX/`. Documentation can be found at `https://sol.physik.hu-berlin.de/cell`. It includes installation instructions, the API, and tutorials. Jupyter notebooks and Python scripts for reproducing the CE of O-Pt/Cu(111) of Sec. 3 are available on Github (`git@github.com:srigamonti/optcu.git`). The *ab initio* data for Si-Ge (Sec. 5.1) and $Ba_8Al_xSi_{46-x}$ (Sec. 5.2) are available in NOMAD [56], DOI `https://dx.doi.org/10.17172/NOMAD/2023.10.24-3` and DOI `https://dx.doi.org/10.17172/NOMAD/2023.10.24-1`, respectively.

Acknowledgements

We thank Luca Ghrinighelli for fruitful discussions on model selection and sampling methods and Matthias Scheffler for drawing our attention to the Wang-Landau method. This work received partial funding from the German Research Foundation (DFG) through the CRC 1404 (FONDA), project 414984028, and the NFDI consortium FAIRmat, project 460197019; the Max

Planck Research Network BiGmax, and the European Union's Horizon 2020 research and innovation program under the grant agreement Nº 951786 (NOMAD CoE).

References

[1] J. W. D. Connolly and A. R. Williams, Phys. Rev. B **27**, 5169 (1983).

[2] J. Sanchez, F. Ducastelle, and D. Gratias, Physica A: Statistical Mechanics and its Applications **128**, 334 (1984).

[3] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual* (CreateSpace, Scotts Valley, CA, 2009).

[4] "`CELL` documentation," `https://sol.physik.hu-berlin.de/cell`.

[5] A. Zunger, S.-H. Wei, L. G. Ferreira, and J. E. Bernard, Phys. Rev. Lett. **65**, 353 (1990).

[6] A. van de Walle and G. Ceder, Journal of Phase Equilibria **23**, 348 (2002).

[7] T. Mueller and G. Ceder, Phys. Rev. B **82**, 184107 (2010).

[8] G. Van Rossum and F. L. Drake Jr, *Python tutorial* (Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995).

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, Journal of Machine Learning Research **12**, 2825 (2011).

[10] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning* (2013) pp. 108–122.

[11] A. H. Larsen, J. J. Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dułak, J. Friis, M. N. Groves, B. Hammer, C. Hargus, E. D. Hermes, P. C. Jennings, P. B. Jensen, J. Kermode, J. R. Kitchin, E. L. Kolsbjerg, J. Kubal, K. Kaasbjerg, S. Lysgaard, J. B. Maronsson, T. Maxson, T. Olsen, L. Pastewka, A. Peterson, C. Rostgaard, J. Schiøtz, O. Schütt, M. Strange, K. S. Thygesen, T. Vegge, L. Vilhelmsen, M. Walter, Z. Zeng, and K. W. Jacobsen, Journal of Physics: Condensed Matter **29**, 273002 (2017).

[12] M. Borg, C. Stampfl, A. Mikkelsen, J. Gustafson, E. Lundgren, M. Scheffler, and J. N. Andersen, ChemPhysChem **6**, 1923 (2005).

[13] J. Sanchez, F. Ducastelle, and D. Gratias, Physica A: Statistical Mechanics and its Applications **128**, 334 (1984).

[14] A. van de Walle, CALPHAD: Computer Coupling of Phase Diagrams and Thermochemistry **33**, 266 (2009).

[15] C. Wolverton and D. de Fontaine, Phys. Rev. B **49**, 8627 (1994).

[16] M. P. Deisenroth, A. A. Faisal, and C. S. Ong, *Mathematics for Machine Learning* (Cambridge University Press, 2020).

[17] S. Foucart, *A mathematical introduction to compressive sensing / Simon Foucart ; Holger Rauhut*, Applied and numerical harmonic analysis (New York [u.a.], 2013).

[18] L. J. Nelson, G. L. W. Hart, F. Zhou, and V. Ozoliņš, Phys. Rev. B **87**, 035125 (2013).

[19] S. Arora, *Computational complexity : a modern approach / Sanjeev Arora ; Boaz Barak* (Cambridge [u.a.], 2009).

[20] R. Tibshirani, Journal of the Royal Statistical Society. Series B (Methodological) **58**, 267 (1996).

[21] L. M. Ghiringhelli, J. Vybiral, E. Ahmetcik, R. Ouyang, S. V. Levchenko, C. Draxl, and M. Scheffler, New Journal of Physics **19**, 023017 (2017).

[22] F. R. Lucci, T. J. Lawton, A. Pronschinske, and E. C. H. Sykes, The Journal of Physical Chemistry C **118**, 3015 (2014).

[23] K. Frey, D. J. Schmidt, C. Wolverton, and W. F. Schneider, Catal. Sci. Technol. **4**, 4356 (2014).

[24] D. J. Schmidt, W. Chen, C. Wolverton, and W. F. Schneider, Journal of Chemical Theory and Computation **8**, 264 (2012).

[25] K. Jacobsen, P. Stoltze, and J. Nørskov, Surface Science **366**, 394 (1996).

[26] L. M. Herder, J. M. Bray, and W. F. Schneider, Surface Science **640**, 104 (2015).

[27] A. H. Nguyen, C. W. Rosenbrock, C. S. Reese, and G. L. W. Hart, Phys. Rev. B **96**, 014107 (2017).

[28] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing, in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, edited by F. Loizides and B. Schmidt (IOS Press, 2016) pp. 87 – 90.

[29] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer Series in Statistics (Springer New York Inc., New York, NY, USA, 2001).

[30] M. Troppenz, S. Rigamonti, and C. Draxl, Chemistry of Materials **29**, 2414 (2017).

[31] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, The Journal of Chemical Physics **21**, 1087 (1953).

[32] W. K. Hastings, Biometrika **57**, 97 (1970).

[33] F. Wang and D. P. Landau, Phys. Rev. Lett. **86**, 2050 (2001).

[34] D. P. Landau, S.-H. Tsai, and M. Exler, American Journal of Physics **72**, 1294 (2004).

[35] A. van de Walle and G. Ceder, Rev. Mod. Phys. **74**, 11 (2002).

[36] M. Tuckerman, en*Statistical mechanics: Theory and molecular simulation*, Oxford Graduate Texts (Oxford University Press, London, England, 2010).

[37] S. N. Khan and M. Eisenbach, Phys. Rev. B **93**, 024203 (2016).

[38] M. Troppenz, S. Rigamonti, J. O. Sofo, and C. Draxl, Phys. Rev. Lett. **130**, 166402 (2023).

[39] A. Qteish and R. Resta, Phys. Rev. B **37**, 6983 (1988).

[40] S. de Gironcoli, P. Giannozzi, and S. Baroni, Phys. Rev. Lett. **66**, 2116 (1991).

[41] B. Dünweg and D. P. Landau, Phys. Rev. B **48**, 14182 (1993).

[42] M. Laradji, D. P. Landau, and B. Dünweg, Phys. Rev. B **51**, 4894 (1995).

[43] K. F. Garrity, Phys. Rev. B **99**, 174108 (2019).

[44] P. Hohenberg and W. Kohn, Phys. Rev. **136**, B864 (1964).

[45] W. Kohn and L. J. Sham, Phys. Rev. **140**, A1133 (1965).

[46] V. Blum, R. Gehrke, F. Hanke, P. Havu, V. Havu, X. Ren, K. Reuter, and M. Scheffler, Computer Physics Communications **180**, 2175 (2009).

[47] F. Knuth, C. Carbogno, V. Atalla, V. Blum, and M. Scheffler, Computer Physics Communications **190**, 33 (2015).

[48] J. P. Perdew, A. Ruzsinszky, G. I. Csonka, O. A. Vydrov, G. E. Scuseria, L. A. Constantin, X. Zhou, and K. Burke, Phys. Rev. Lett. **100**, 136406 (2008).

[49] G. L. W. Hart and R. W. Forcade, Phys. Rev. B **77**, 224115 (2008).

[50] J. P. Dismukes, L. Ekstrom, and R. J. Paff, The Journal of Physical Chemistry **68**, 3021 (1964).

[51] T. Vogel, Y. W. Li, T. Wüst, and D. P. Landau, Journal of Physics: Conference Series **487**, 012001 (2014).

[52] P. C. Kelires and J. Tersoff, Phys. Rev. Lett. **63**, 1164 (1989).

[53] G. J. Snyder and E. S. Toberer, Nature Materials **7**, 105 EP (2008).

[54] J. Brorsson, A. E. C. Palmqvist, and P. Erhart, The Journal of Physical Chemistry C **125**, 22817 (2021).

[55] A. Gulans, S. Kontur, C. Meisenbichler, D. Nabok, P. Pavone, S. Rigamonti, S. Sagmeister, U. Werner, and C. Draxl, Journal of Physics: Condensed Matter **26**, 363202 (2014).

[56] C. Draxl and M. Scheffler, Journal of Physics: Materials **2**, 036001 (2019).