

XTALOPT Version 14: Variable-Composition Crystal Structure Search for Functional Materials Through Pareto Optimization

Samad Hajinazar^a and Eva Zurek^{a,*}

^a*Department of Chemistry, State University of New York at Buffalo, Buffalo, New York 14260-3000, United States*

*Corresponding author: ezurek@buffalo.edu

Abstract

Version 14 of `XtalOpt`, an evolutionary multi-objective global optimization algorithm for crystal structure prediction, is now available for download from its official website <https://xtalopt.github.io>, and the Computer Physics Communications Library. The new version of the code is designed to perform a ground state search for crystal structures with variable compositions by integrating a suite of *ab initio* methods alongside classical and machine-learning potentials for structural relaxation. The multi-objective search framework has been enhanced through the introduction of Pareto optimization, enabling efficient discovery of functional materials. Herein, we describe the newly implemented methodologies, provide detailed instructions for their use, and present an overview of additional improvements included in the latest version of the code.

1 Introduction

The development of computational methodologies and tools for the discovery of novel functional materials has taken center stage with the advent of diffusion-based models to generate crystal lattices [1, 2, 3, 4], machine learning (ML) interatomic potentials to optimize these lattices and compute their energies [5], as well as ML-based models for predicting material's properties [6, 7, 8, 9, 10, 11, 12, 13]. More traditional crystal structure prediction (CSP) methods for materials discovery [14, 15, 16, 17, 18] have also benefited from the myriad advances in artificial intelligence for materials. In a typical CSP search, the potential energy surface of the target chemical system has traditionally been mapped through *ab initio* calculations or classical force fields, and – more recently – using ML potentials [19, 20, 21, 22, 23, 24, 25]. Then, global optimization techniques are utilized to navigate this energy landscape, while the energetics of the examined structural motifs are employed to find possible routes towards more stable phases.

Various global optimization strategies have been implemented within commonly used software platforms for CSP, such as evolutionary algorithms (`XtalOpt` [26, 16], `MAISE` [27], `GASP` [28], and `USPEX` [29]), particle swarm optimization (`CALYPSO` [30]), and random structure searching (`AIRSS` [31]). We have previously illustrated that the commonly used evolutionary algorithm is likely to have a higher success rate of finding the ground state structure in fixed-composition searches as compared to a purely random search [16]. However, employing constraints based upon chemical and physical intuition has made random structure searching (e.g., as implemented in the `AIRSS` code) a very powerful technique. Further extension of the basic search algorithms to multi-objective global optimization (MOGO) approaches [32, 33] has facilitated the ground state search for functional materials [34, 35, 36, 37, 38, 39, 40, 41, 42, 43].

Another key direction in addressing the technological demand for new materials through CSP approaches involves enabling the optimization algorithm to search the composition space of a desired chemical system, i.e., a “variable-composition” (VC) ground state search [44, 45, 46, 47, 48, 49, 50, 51]. In particular, the advent and rapid advancements of ML interatomic potentials, including the more recent family of universal interatomic potentials (UIPs) [52, 53, 54, 55, 56, 57, 58], offers an enormous new opportunity for high-throughput exploration of various chemical systems in searching for (meta)stable phases of novel functional materials.

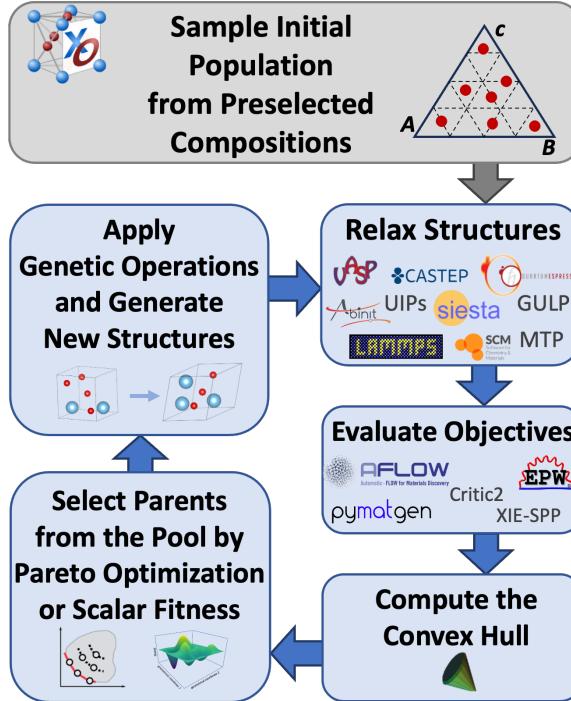


Figure 1: The workflow in **XtalOpt** version 14, which offers a variable-composition multi-objective (VC-MO) search. Post relaxation, each structure’s energies are used to obtain its distance above the convex hull, which is combined with the values of the user-specified objectives to determine the structure’s suitability for parenting new offspring. Genetic operations are then applied to the selected parents to create new structures with compositions determined by the search type.

XtalOpt, an evolutionary algorithm designed for CSP, is a fully open-source and cross-platform software package, available in both command-line interface (CLI) and graphical user interface (GUI) modes. It is capable of performing both single-objective (SO) and multi-objective (MO) searches [59] for the ground state structures of crystalline systems. Originally designed to search over a fixed chemical composition, previous implementations of **XtalOpt** [26, 60, 61, 62, 16, 59] start by producing an initial set of structures generated from specified supercells of the given empirical formula. Following successful local optimization of the population members, the selection of parent structures was based on the fitness, determined from the normalized relative enthalpies obtained in an SO search, or normalized relative value of user-defined objectives in a MO search [43, 62, 63, 64]. The offspring would be created from the parent structure(s), with the size and composition matching those of the initial user-defined list, through application of genetic operations. These genetic operations included: “crossover” where two parent structures are mixed to generate an offspring with the same composition, as well as “stripple” and “permustrain” mutations that generate a new structure by applying random distortions to a selected parent structure. Filtration of the parent pool was introduced to keep only those structures with desirable characteristics [65].

In the present work, we introduce the new release of **XtalOpt** (version 14) where, among various improvements, the MOGO functionality is extended by including the Pareto optimization algorithm, implementing a VC search capability that is natively integrated with the MOGO, and easy-to-use interface scripts are included for the utilization of modern UIPs in structural relaxation. These developments therefore pave the way towards an efficient ground state search for crystalline phases of materials with variable composition and desired target properties.

The workflow of the **XtalOpt** VC search (optionally with multiple objectives) is illustrated schematically in Fig. 1. The following steps are taken: (i) the initial population of structures is generated from a set of user-defined compositions and (optionally) using the `randSpg` library [66], (ii) the population members

are locally optimized using one of the supported platforms, (iii) the user-specified objectives (if any) are calculated, (iv) the convex hull for the entire population is calculated and the distance above the convex hull for all successfully relaxed structures is updated via the `Qhull` library [67], (v) the parents for new structures are selected from the pool either through Pareto optimization or by using the generalized scalar fitness function, (vi) new offspring are produced from the selected parent structures, and they are submitted for local optimization. This cycle is repeated until a pre-specified number of structures are optimized.

The implementation of the VC search functionality is made possible through a series of modifications to the `XtalOpt` algorithm, namely, using the convex hull of the population to obtain the energetic fitness of structures [44], expanding the existing genetic operations to perform evolutionary transformations on parent structures of (possibly) different composition and producing an offspring whose composition matches the desired search type, and adjusting the workflow for generating high-symmetry unit cells, or those that involve molecular units, etc.

The details of the implemented methodologies and instructions for using them are discussed in the following sections. While most of the described options are available in version 14.0 of `XtalOpt`, the multi-cut crossover functionality (Section 5.1) and the option to set a minimum total atom count (Section 4) are introduced in versions 14.1 and 14.2, respectively.

2 Input chemical formulas

In the previous versions of `XtalOpt` [26, 59], the search over a fixed chemical composition was initiated with specifying the chemical system by its reduced empirical formula, e.g.,

```
empiricalFormula = Ti1O2
```

over an optionally determined list of formula units, e.g.,

```
formulaUnits = 1 - 4
```

In the new version of `XtalOpt`, introducing the chemical system and (optionally) desired formula units differs in the following ways from the previous versions: (i) the “`empiricalFormula`” and “`formulaUnits`” pair of flags is removed, and (ii) the chemical system is now entirely specified by the entries of a single input flag “`chemicalFormulas`”.

Generally, an input entry for “`chemicalFormulas`”:

1. Is interpreted as the “explicit” formula of the simulation unit cell (instead of the reduced empirical formula), and
2. Must include the “full chemical formula” (i.e., a combination of element symbols and corresponding atom counts, e.g., “Ti1O4” instead of “TiO4”).

That is to say, the input:

```
chemicalFormulas = Ti4O8
```

instructs `XtalOpt` to search in the Ti-O system with “1:2” composition, while all generated unit cells have four Ti atoms and eight O atoms.

Various “formula units” of a chemical system can be introduced using a comma-delimited list of full chemical formulas, e.g.,

```
chemicalFormulas = Ti1O2, Ti2O4, Ti3O6, Ti4O8, Ti8O16
```

Besides explicitly listing various formula units in the chemical formulas input, they can also be combined into a hyphen-separated list of formulas in the form of a “*formula1 - formula2*” entry. In such an entry, both ends must be full chemical formulas of the same composition, while “*formula2*” is a proper supercell of “*formula1*”. That is to say, the number of atoms of each element type in “*formula2*” must be a fixed integer multiple of those in “*formula1*” (e.g., “A3-A7” does not work, while “A3-A6” is accepted).

As an example, the above input can be specified using a hyphen-separated entry as follows:

```
chemicalFormulas = Ti1O2 - Ti4O8, Ti8O16
```

The above input corresponds to specifying the pair of input flags in the previous versions of the code as: “empiricalFormula = Ti1O2” and “formulaUnits = 1-4, 8”.

The “chemicalFormulas” parameter not only determines the initial cell(s) to be created in the first generation, it also has an important role in specifying what type of the evolutionary search should be conducted, as detailed in the next section.

3 Various types of evolutionary searches

Generally, the overall search workflow of the new version of **XtalOpt** is similar to the previous versions: an initial set of structures is generated from the compositions and cell sizes specified in the input, and the genetic operations are applied to the selected parents from the pool to produce offspring. However, now, the details of how the genetic operations make new offspring and the children that can be created are different. That is to say, depending on the search type, the size and composition of the offspring produced by genetic operations can be different than that of their parent structure(s) or the input chemical formula list. The search type is determined by the details of the input formulas list and a set of new flags, as described in the next subsections.

3.1 Fixed-composition search

In the simplest case, if all the input formulas are from the same composition, the search is a “fixed-composition” (FC) evolutionary search, where the offspring have their composition match that of their parent structure(s) (i.e., initial list). This is essentially the traditional search performed by previous versions of **XtalOpt**.

3.2 Multi-composition search

In the new version of **XtalOpt**, on the other hand, the initial list of chemical formulas can also include a combination of different compositions, e.g.,

```
chemicalFormulas = Ti2O3, Ti1O2 - Ti4O8, Ti5O3, Ti1O1 - Ti5O5
```

This input instructs **XtalOpt** to perform a “multi-composition” (MC) search. This type of search is similar to the FC search in that the initial set of structures is generated as usual, and subsequent individuals are forced to have a composition matching “one of the compositions specified in the input list”. Since the initial list now covers various compositions, however, the parent structure(s) can be chosen from different compositions. This extends “desired” motifs found in one composition to other ones, potentially speeding up the search in finding the best candidate structures for various compositions.

3.3 Variable-composition search

Regardless of whether the input formulas include one or more compositions, if the input flag

```
vcSearch = true
```

is specified, after the first generation is produced (as usual and according to the input formulas list), the genetic operation crossover is now allowed to produce cells with new compositions that do not necessarily match their parent structures' composition, or even those compositions listed among the initial chemical formulas. This so-called “variable-composition” (VC) search is the most general type of evolutionary search conducted by **XtalOpt**. In it the entire chemical system of the specified elements is explored to find (meta)stable phases of various compositions.

4 Minimum and maximum number of atoms

Version 14 of **XtalOpt** can generate structures with new compositions (hence, various numbers of atoms in the cell) and random supercells (discussed below). To control the computational cost of the search an input parameter, “maxAtoms”, is introduced to set the maximum number of atoms in the generated unit cells during the search. A new structure in the VC search or a random supercell will be generated only if the total number of atoms in the final cell complies with this set limit.

The default value for the maximum number of atoms is 20. However, and regardless of being specified in the input or having its default value, if any entry in the input chemical formulas list has a larger total number of atoms, **XtalOpt** will re-adjust this parameter automatically to match the largest cell size in the input.

In addition to the maximum atom counts setting, the user can set a lower limit for the total atom counts in generated unit cells through the “minAtoms” flag. This parameter, which is relevant to a VC search, has the default value of 1. If it is explicitly initialized to a value larger than the smallest atom counts in the input chemical formulas, **XtalOpt** produces a warning message and sets this parameter to the lowest possible value, consistent with the input formulas.

5 Genetic operations

XtalOpt uses various genetic operations to produce new structures from selected parent structure(s). In all of the previous versions, the defined operations included crossover, strippling, and permustrain [26]. Each operation would be chosen based on their user-defined “percent chance”. The specified chances for all operations, each an integer in the [0,100] range, was required to sum to 100.

In **XtalOpt** version 14, the crossover operation is extended by allowing for “multi-cut” merging of the parent cells, and two new genetic operations (applicable only to the VC search) and a new random mutation (relevant to all types of evolutionary searches), are introduced. Moreover, the input entries defining the probability of choosing the various genetic operations and the interpretation of their values have been changed, as described below.

5.1 Multi-cut crossover

Prior to **XtalOpt** version 14, the crossover operation was performed by a single-point “cut and splice” of the parent structures [26]. The user now has the option to instruct the code to cut the parent unit cells at multiple points to produce “ribbons” by using the input flag:

```
crossoverCuts = 3 # default is 1
```

Generally, if this flag is set to “ n ”, the code will cut each parent structure in “ $n+1$ ” ribbons, and will select the atoms of the offspring structure from the alternating ribbons of the parent structures. The width of these ribbons is initialized equally, and then distorted randomly. For the case of the single-cut crossover, however, the setting for the minimum contribution of each parent is applied. By default, **XtalOpt** will perform a single-cut crossover, unless the user instructs otherwise. This setting can be modified during the run. It should be noted that specifying an “even” number of cut points will result in an “odd” number of ribbons, hence, an uneven selection of atoms for the produced structure.

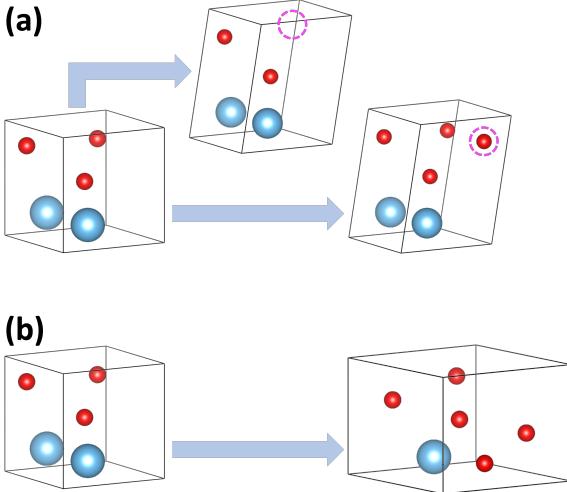


Figure 2: Schematic illustration of the new genetic operations implemented in **XtalOpt** version 14, applicable only to the VC search. (a) Permutomic produces an offspring by randomly adding or removing an atom to the parent structure, followed by a small random lattice distortion. (b) Permucomp is designed to diversify the population through producing a new structure by distorting the parent lattice and decorating it with a new random composition.

5.2 New genetic operation: permutomic

To ensure full sampling of the stoichiometry space, a new evolutionary operation, “permutomic”, has been introduced in the VC search. This operation randomly adds (removes) an atom to (from) a structure chosen from the parent pool, as schematically illustrated in Fig. 2(a). For such a search, the user should provide a probability for performing this operation in the input. This probability is adjustable during the run.

5.3 New genetic operation: permucomp

Another evolutionary operation introduced in the VC search to facilitate the exploration of various stoichiometries is “permucomp”. This operation, illustrated in Fig. 2(b), creates a new composition with a randomly chosen total number of atoms (up to “maxAtoms” parameter).

This random mutation is primarily designed to diversify the structure pool in long searches over multiple elements. Since it has a limited chance of yielding an energetically favorable structure, ideally it should be assigned a relatively small probability compared to other genetic operations.

5.4 Chances of performing the genetic operations

In **XtalOpt** version 14, instead of using a “percent chance” for applying the various genetic operations, the probability of choosing a particular operation should be specified by a “relative weight”. In the CLI input the relative weights can be specified using the following set of flags (instead of the “percentChances...” flag):

```
weightCrossover
weightStripple
weightPermustrain
weightPermutomic
weightPermucomp
```

The weight for each operation should be zero or any positive integer, and it can be modified during the run. It should be noted that no specific condition needs to be satisfied for the total sum of the weights (i.e. the sum does not need to equal 100).

Input flag	Default value	Percent chance	
		VC	FC/MC
weightCrossover	35	33%	40%
weightStripple	25	24%	30%
weightPermustrain	25	24%	30%
weightPermutomic	15	14%	0%
weightPermucomp	5	5%	0%

Table 1: Input flags to specify the relative weights of various genetic operations, with their default values. The actual percentage chances of applying a genetic operation are listed in the last columns for the VC and FC/MC search types.

Assuming that the set of relative weights of the genetic operations are given by $\{P_i\}$, `XtalOpt` determines the percent chance of applying the j^{th} genetic operation, C_j , as:

$$C_j = 100 \times \frac{P_j}{\sum_{i=\text{all relevant operations}} P_i}$$

The sum over relative weights includes genetic operations relevant to the search: for a VC search it includes all five genetic operations, while for a non-VC search the weights of permutomic and permucomp are ignored and their chances are set to zero.

The default values for the relative operation weights, and the (approximate) corresponding runtime percent chance for applying them for the various search types are summarized in Table 1.

If all the relevant genetic operations for a specific type of search have zero weight in the input file, `XtalOpt` will use an equal probability for applying the genetic operations, while a warning message is printed in the run output.

It should be noted that our benchmarking tests performed on the Ti-O system show that the number of unique chemical compositions produced in a VC search increases by 30% or more with the use of permutomic and permucomp operations, with the default genetic operation weights.

5.5 Secondary mutation: random supercell expansion

Although not a genetic operation *per se*, the user can optionally define a finite probability as a percent chance (in the [0, 100] range), by which the structure generated from any of the existing genetic operations will be expanded to a supercell. This probability is zero by default, and can be set using the input flag:

```
randomSuperCell = 0
```

If this flag is specified to be a non-zero value, the expansion is performed by randomly choosing three integers as multiples for the lattice vectors, such that the produced supercell has up to “maxAtoms” total atoms in the cell. Further, a randomly chosen atom in the supercell is displaced randomly, subject to the minimum interatomic distances settings (Fig. 3). This operation can be useful in ground state searches where charge density waves lower the energy of the system.

6 Reference energies

As of version 14, `XtalOpt` uses the distance above the convex hull as the target value for global energy minimization, as first introduced by Zunger and co-workers [44]. The calculation of the convex hull (performed using the `Qhull` library [67]), requires defining the reference energies. By default, the code uses “0.0” as the “elemental reference energies”. To obtain the correct convex hull for MC and VC searches, however, the correct reference energies need to be specified.

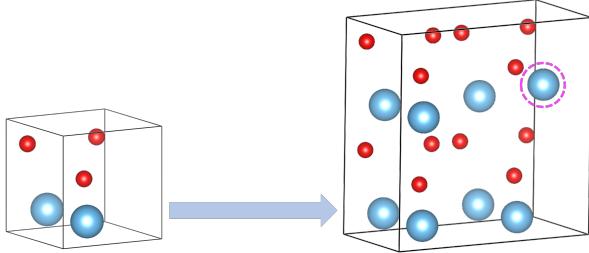


Figure 3: The secondary mutation “random supercell expansion” is applicable to the offspring produced by any of the `XtalOpt` genetic operations, and produces a random supercell in which an atom is randomly selected and displaced. The default probability of this mutation is 0%.

The user can define reference energies in the input through a comma-delimited list in which each entry includes a full chemical formula followed by the corresponding total energy (or enthalpy), e.g., for the Ti-O system:

```
referenceEnergies = 01 -2.53, Ti4 -17.65
```

The energies introduced in the input must have the same unit as is used by the local optimizer (e.g. meV, eV or Hartree) for the chemical formula listed to produce the correct results.

Furthermore, the user can introduce reference energies not only of the elements but also for relevant multi-element (sub)systems if they are needed for calculating the correct convex hull. An example of where this approach may be useful is if a particular binary or ternary is known to lie on the hull. For example, for the O-Ti-N system, this can be a valid entry:

```
referenceEnergies = 02 -5.06, Ti2 -8.825, N1 -3, O1Ti1 -6.32, O2N5 -12.7, Ti1N2O1 -5.80
```

It should be noted that if any reference energy is introduced, for consistency, all elemental reference energies must be specified. Otherwise `XtalOpt` issues an error message and quits.

7 Pareto optimization and parent selection

In release 13 of `XtalOpt` [59] a basic generalized fitness function, calculated as the weighted sum of normalized values of objectives, was introduced to perform a MOGO search for materials with desired functionalities. The effectiveness of this formalism has been demonstrated through powder X-ray diffraction-assisted discovery of meta-stable phases of Ti-O [43], and the search for high-pressure electride phases of Ca_5Pb_3 using electronic charge density information [68]. Further, it was previously shown that filtering the parent pool based upon structural features that include the local crystalline order and symmetry is an effective way to find metastable phases [65].

By default `XtalOpt` employs this scalar fitness function for global optimization (i.e., the “Basic” optimization type) regardless of the number of objectives. In the new version of `XtalOpt` Pareto optimization, which is in principle applicable to both SO and MO searches, has been implemented as an alternative (the “Pareto” optimization type). The user can instruct the code to utilize Pareto optimization through the input flag:

```
optimizationType = Pareto # default is Basic
```

For Pareto optimization, `XtalOpt` uses the NSGA-II algorithm [69] in which the Pareto front (rank) and crowding distances of the structures in the parent pool are employed to select a new parent through a binary tournament selection.

The workflow of selecting a new parent structure in Pareto optimization includes: (i) performing a standard non-dominated sorting to identify the Pareto front (rank) of all candidate structures, (ii) calculating crowding distances for structures in each front, (iii) randomly selecting a pair of structures from the pool, and (iv) choosing the parent structure that has a better rank, or a larger crowding distance (if both structures belong to the same Pareto front) from the selected pair. For structures with a similar rank and crowding distance, the selection is made randomly.

Since **XtalOpt** is a population-based algorithm, the parent pool includes all successfully optimized structures, hence, elitism in the Pareto optimization is automatically maintained. Nevertheless, the tournament selection is made within the entire population of locally optimized structures after similar or duplicate (see Sec. 9) structures are identified, unlike the scalar fitness (basic) scheme where a limited subset of the top candidate structures (i.e., equal to the size of the parent pool specified by the user) are considered for randomly selecting a parent structure. The user, however, has the option to restrict the tournament selection to the user-specified pool size by the flag:

```
restrictedPool = true # default is false
```

If the parent pool is restricted for tournament selection in Pareto optimization, the code will sort the structures in the entire pool according to their rank (and then according to the crowding distances within each rank), and select the top candidates such that there will be as many as the user-specified parent pool size from which the tournament selection is performed.

Besides tournament selection, **XtalOpt** provides another parent selection option within the Pareto scheme. Here, the obtained rank and crowding distances of the structures are used to calculate a scalar fitness, employed to determine the parents, similar to the case of basic optimization. This can be invoked by using the flag:

```
tournamentSelection = false # default is true for Pareto optimization
```

The Pareto-based fitness value relies on the calculated rank of structures, r , which is a value in the $[0, N - 1]$ range, where N is the total number of Pareto fronts and $r = 0$ corresponds to the global Pareto front. Using the obtained ranks, solutions of each front are assigned a raw fitness value of

$$f_r^0 = \frac{N - r}{N} \quad (1)$$

that corresponds to a fitness of 1 for structures on the global Pareto front and $\frac{1}{N}$ for those on the last Pareto front. Next, the calculated crowding distances of structures in each front are scaled to the $[0.1, 1]$ range, where the non-zero lower limit is chosen to eliminate the overlap between solutions of two successive fronts.

Finally, the Pareto-based fitness for the i^{th} structure in the r^{th} Pareto front with a scaled crowding distance of $d_{r,i}$ is calculated using the following formula:

$$f_{r,i} = f_r^0 - \left(\frac{1 - d_{r,i}}{N} \right) \equiv f_{r+1}^0 + \frac{d_{r,i}}{N} \quad (2)$$

The above procedure is designed to: (i) ensure the rank-precedence by introducing a gap of at least $\frac{0.1}{N}$ between the “worst” solution of rank r and the “best” solution of rank $r + 1$, (ii) diversify the selection pool by prioritizing more unique candidate structures in each rank through applying the crowding distance values, and (iii) obtain a scalar fitness value in the range of $[0, 1]$.

As outlined above, by default, **XtalOpt** applies crowding distances in Pareto optimization in both tournament selection and Pareto-based fitness calculation. This can be modified by specifying the input flag:

```
crowdingDistance = false # default is true
```

which results in performing tournament selection only using the structural ranking, or calculating Pareto-based scalar fitness with $d_{r,i} = 1$ for all structures (i.e., the raw fitness values obtained from Eq. 1).

It is important to note that the outcome of non-dominated sorting is especially sensitive to the numerical precision of the objective values. Although **XtalOpt** does not manipulate the calculated values of the objectives, it provides the user with the option to apply a desired precision to them prior to the parent selection process. This may be done by specifying the number of decimal digits to be retained using the flag:

```
objectivePrecision = 6 # default is -1
```

where the default value of “-1” implies not rounding the objective values.

For an MO search performed using the previous version of **XtalOpt** [59], specifying the objective weight and output file name for the executable script was optional in the CLI mode. In the new version of the code, these fields are all mandatory. Therefore, all four input entries for an objective must be given in the below order:

```
objective = objective_type executable_script_path output_filename weight
```

Furthermore, the weight for a “Filtration” objective must be set to zero in the input.

It should be noted that:

- While the weights are not used for the Pareto optimization, they must always be specified since the user has the option to change the optimization type during runtime (Pareto to basic, and vice versa), and
- Similar to the previous workflow, any minimizable/maximizable objective with a weight of zero will be calculated but will not be considered in the scalar fitness for the basic optimization scheme.

8 Volume limits

In the previous versions of **XtalOpt**, the volume of the unit cell could be constrained by specifying either the absolute minimum and maximum limits (in Å³ per formula unit), or the scaled volume limits.

In **XtalOpt** version 14, the absolute volume limits are read in “Å³ per atom” units and the corresponding flags are renamed to:

```
minVolume  
maxVolume
```

Also, the pair of flags for introducing the scaled volume limits are renamed to:

```
minVolumeScale  
maxVolumeScale
```

To be consistent with the minimum atomic radii settings, the scaled volume limits in the new version of **XtalOpt** are considered as factors of spheres of “covalent” radii of atoms (whereas the previous version of the code used van der Waals radii). The scaled volume settings of the previous version, as a result, need to be multiplied by ~1.5 to obtain similar results in the new version.

Moreover, in the new version of **XtalOpt**, the user has the option to define volume limits for the elements by providing a list of comma-delimited entries. This setting will only take effect if the limits are defined for all elements in the chemical system. Each entry includes a full chemical formula for the elemental unit cell followed by the corresponding volume limits (minimum and maximum, respectively) in units of Å³, e.g.,

```
elementalVolumes = 01 20 40 , Ti2 50 100
```

Generally, the volume constraints in `XtalOpt` are applied in the following order:

1. If elemental volumes are given (properly), they are used first,
2. If scaled factors are given (properly), they are used,
3. If explicit volume limits (per atom) are given, they are used,
4. If none of the above, the default absolute volume limits (1 and 100 Å³ per atom) are used.

9 Similarity check

Given that hundreds to thousands of structures are often optimized in a single evolutionary run, many of them end up being duplicates of one another. Removal of such duplicates from a search is of prime importance, otherwise the run can be biased by a particular structural motif, failing to explore the relevant areas of the potential energy landscape. However, duplicate identification is hampered by the fact that many structures are not fully optimized and because a single structure can be represented by different cell shapes and sizes.

The first version of `XtalOpt` used an approximate scheme, identifying duplicates by comparing their space-groups, volumes and enthalpies [26] to within a user-specified tolerance. Because this method was prone to false positives, an exact comparison technique that searches for transformations mapping atoms from one structure to another was developed. The resulting `XtalComp` library [70] was used to find duplicates since version 9 [61] of `XtalOpt`. Duplicates were marked as such in the output and removed from the run. Though this exact matching scheme was key for finding certain inorganic compounds [71], it is dependent upon the choice of the user-specified tolerance and does not work well for molecular systems. For example, it would likely fail to flag structures that differ by a slight rotation of a molecule or a functional group as duplicates because the required tolerance would be too large.

In these cases approximate methods that employ radial distribution functions (RDF) or atomic separation metrics to identify similar structures may be useful [72, 73, 74]. Therefore, we have implemented a similarity check, which uses the RDF of species-resolved bonds to detect similarities between structures as first introduced in the `MAISE` code [27], in `XtalOpt` version 14. Similar structures are labeled as such in the output. For instance, a label of “*Sim(2×10)*” for the status of a structure in the `results.txt` file means that this structure is similar to the structure “*2×10*”, within the given tolerances for the similarity check.

This functionality is controlled by a numerical value, in the [0,1] range, set by the input flag:

```
rdfTolerance = 0.95
```

If the user specifies a value greater than zero for the RDF tolerance, `XtalOpt` will use the RDF similarity check instead of `XtalComp`. Specifically, it calculates the scalar product of normalized RDF vectors of structures, and those that have a value larger than the specified tolerance will be marked as “similar”.

By default, the RDF tolerance is set to zero; hence `XtalComp` is employed for duplicate checking to maintain consistency with previous versions of `XtalOpt`. In general, however, the use of RDF check is recommended, as it provides a quantitative measure for the similarity of structures that can be visualized using common plotting software.

The details of the RDF vector calculations depend on a set of parameters, i.e.,

- Cutoff value for the included bond length (in Å),
- Spread of Gaussian functions used for smoothing the bond length distribution (in Å),
- Number of bins (over the bond length range of [0, cutoff]) for sampling the distribution.

In the CLI mode, this can be done through the following input flags (with their default values):

```
rdfCutoff = 6.0
rdfSigma = 0.008
rdfNumBins = 3000
```

10 Seed structures

In previous versions of `XtalOpt` seed structures could be introduced using a space-separated list. Now, for consistency, a comma-delimited list of entries should be used instead, e.g.,

```
seedStructures = /path1/POSCAR1,/path2/POSCAR2
```

In previous versions the seeds were required to possess the same composition as the reference chemical system of the search. However, in the new version of `XtalOpt` a seed structure can be an “off-composition” structure, i.e., a structure with a composition that is not found in the input formulas list or is a sub-system of the reference chemical system, such as binary or elemental structures in a ternary search.

The off-composition seeds will be read in as long as they do not include any element that is not in the reference chemical system. They will be included in the parent pool upon successful local optimization, and can participate in the genetic operations if selected as a parent structure (depending on their fitness value). However, since the stripple and permutstrain operations are not designed to manipulate the parent structure’s composition, they will not be applied to an off-composition seed structure in the FC and MC searches.

One example where providing sub-system seeds could be useful is for obtaining energies of reference structures, to be subsequently used in the convex hull construction, that are consistent with the local optimization settings of the search. This option, however, should be utilized carefully since a failure of the local relaxation of the reference structure by the external code will result in an incorrect convex hull leading to errors in fitness determinations. The introduction of sub-systems may also be useful if they contain certain structural motifs that could be present in the full system.

The seed structures, in general, are not required to comply with the minimum and maximum number of atom settings. Once a seed structure is read in with a total atom count that is not consistent with the existing settings, the atom counts limits will be updated, while a warning message is produced in the output.

11 Construction of molecular units

For an MC or VC search, it is possible to use molecular units for the construction of the initial random generation of structures, as first introduced in `XtalOpt` version 10 [75] for a FC search. However, now the number of atoms of each type that can participate in the construction of molecular units is limited to the smallest number of atoms of that type in the input list of chemical formulas, regardless of search type.

For instance, given the following input:

```
chemicalFormulas = Ti2O4 - Ti4O8, Ti5O3
```

up to 2 and 3 atoms of Ti and O, respectively, can be used in the construction of molecular units. After initializing the molecular units in a unit cell, the remaining atoms are added randomly to the cell to reach the desired composition.

12 Interface to machine-learning interatomic potentials

XtalOpt explicitly supports a number of optimizers whose output it can parse to find the structural coordinates and energies it requires. A simple way to employ an arbitrary optimizer (e.g., a machine learning interatomic potential) is to use simple scripting to convert its output to that of an already supported optimizer. For instance, if the user sets the optimizer type to VASP while employing an arbitrary code to perform local optimizations, the job file for the XtalOpt run would include the following steps: (i) converting the VASP structure file (POSCAR) to the appropriate format for the optimizer employed, (ii) performing the local optimization, and (iii) extracting the results from the optimizer and writing VASP format output (i.e., OUTCAR and CONTCAR) files.

Such a workflow would allow the user to benefit from the considerable speed-up that is offered by machine learning potentials in an XtalOpt run. This is especially helpful in a VC search for multi-element systems where it might be necessary to explore the entire composition space. Such a run possibly involve thousands of local optimizations, which would be computationally prohibitive using first-principles approaches. Another possibility would be to first relax a structure locally with a machine learning potential, followed by a DFT single point calculation to obtain a more accurate energy prediction.

An easy-to-use Python script, `vasp_uip.py`, is included in the new release of XtalOpt to facilitate such calculations. This wrapper enables local optimizations with a series of the recently developed universal interatomic potentials (UIPs), MACE [52], CHGNet [53], Orb [54], and MatterSim [55], to perform local structure optimization using the Atomic Simulation Environment (ASE) library [76]. The input/output format is that of the VASP code, i.e., the script reads in a POSCAR file and produces a CONTCAR and a minimal OUTCAR file, which follows the VASP format for the outputted entries.

The local optimization specifications and other parameters of this script can be adjusted through a series of command line options. A full list of available options can be obtained by typing:

```
python3 vasp_uip.py -h
```

A simple calculation of the energies, forces, and stresses can be performed with this script as:

```
python3 vasp_uip.py
```

provided a POSCAR file is present in the working directory, and the required libraries are accessible through the invoked python binary.

With the above-mentioned input and output file formats, this script is designed to be used in the XtalOpt runs as a VASP optimizer, allowing for the efficient ground state search for a desired chemical system. It should be noted that the implemented potentials have been chosen as prototypes of the increasingly popular UIPs, and extending the script to support other similar platforms is straightforward.

13 Miscellaneous

13.1 Input entry conventions

In a CLI run, various search settings are provided by the user in the input file, through entries of pre-defined flags. While most flags require a single value as the input entry, sometimes an entry is expected to include multiple parameters (e.g., the “objective” flag). There are also input flags that can parse multiple entries, while the entries might or might not involve multiple parameters (e.g., the “elementalVolumes” and “seedStructures” flags).

In general, and for consistency, the input patterns for the CLI flags are designed with the following conventions:

- When input involves multiple entries, they should be a list of *comma-delimited* entries such as:

```
seedStructures = structure1 , structure2
```

- If a flag requires a single entry, and that entry has multiple parameters, the parameters should be a list of *space-separated* values, e.g., the specification of an objective (which requires four parameters):

```
objective = fil /bin/script output.txt 0.0
```

- If a flag accepts multiple entries that are multi-parameter, combining the above rules, the input should be a *comma-delimited list of space-separated parameters*, e.g., specifying the volumes for multiple elements (with three parameters for each elemental entry):

```
elementalVolumes = Ti1 20 35, O1 15 25
```

The GUI input fields for chemical formulas, reference energies, and elemental volumes should be initialized with input strings adhering to the above format.

13.2 Output files

In `XtalOpt` 14 a new file, `hull.txt`, is created in the local run directory. It includes the composition (atom counts of various elements), total enthalpy, and the calculated distance above the hull for successfully optimized structures. Also included in this file is the Pareto front index, creation index, and a unique tag (generation and id) of each structure.

Moreover, with the exception of the `results.txt` and `xtalopt.state` files, the names of the other output files (produced in the local run directory) are changed in the new version of the code, as listed in Table 2.

Old filename	New filename
<code>xtalopt-runtime-options.txt</code>	<code>cli-runtime-options.txt</code>
<code>xtaloptSettings.log</code>	<code>settings.log</code>
<code>xtaloptDebug.log</code>	<code>output.log</code>

Table 2: Old and new names of output files in an `XtalOpt` run.

13.3 Verbose output

Starting from version 13 of `XtalOpt`, if the code was compiled with the `cmake` flag

```
-DXTALOPT_DEBUG=ON
```

the output of the run would include extra information about the run, and this output would be saved to disk as the `xtaloptDebug.log` file (for both the CLI and GUI modes). In `XtalOpt` release 14, for simplicity and regardless of the compilation flag, the standard output of a CLI run will include an extensive set of additional information (e.g., details of calculating objectives and parent selection probabilities, dot product of the RDF vectors, workflow of genetic operations, etc.) by setting the following input flag:

```
verboseOutput = true # default is false
```

This parameter has a default value of “false”, and it can be modified during the run. It should be noted that, especially for a long search involving large unit cells, this option can produce an enormous output file size; hence, it is most appropriate to be used for debugging purposes.

In the GUI mode, this option is available as a check box in the “Progress” tab (Fig. 4(d)). However, as in the previous version of `XtalOpt`, the `output.log` file (that may or may not include the debug-type additional

information) will be produced only if the code is compiled with the above `cmake` flag. As a result, and depending on the compilation type, this option in the GUI can be either selectable or deactivated.

13.4 Convex hull snapshots

In the CLI mode, if the following flag is set:

```
saveHullSnapshot = true # default is false
```

after each successful local optimization a snapshot of the hull data (i.e., a copy of the `hull.txt` file) will be saved in the local working directory under the folder “movie” as a file named `YYMMDD_HHmmSS_LLL` where: YY (year), MM (month), DD (day), HH (hour), mm (minute), SS (seconds), and LLL (milliseconds) represent a unique and ordered identifier of when the file was written.

This option is specifically designed to facilitate monitoring the progress of the run or to create convex hull images and movies, directly from the file data or using external tools such as the `pycxl` code [77].

In the GUI mode, the same option is available as the “Save hull snapshots” check box under the “Search Settings” tab.

13.5 Computational scaling

In order to measure the computational cost of various `XtalOpt` tasks in a typical search, we performed a series of benchmarking test runs on the Ti-O system to generate 100, 400, and 1000 structures. These searches were conducted over a variety of input parameters, e.g., VC and non-VC searches, basic and Pareto optimization, with and without additional objectives, and using RDF and XtalComp similarity checks. The runs were performed on 24 cores with 20 parallel local optimizations, and the `vasp_uip.py` script with MACE UIP was employed for structural relaxations.

The profiling results were then used to measure the CPU time usage of various `XtalOpt` functions and tasks, and compare them against the overall `XtalOpt` CPU usage and the total CPU time of the run, i.e., the total time consumed by the code, external objective calculations, and the local optimizations that correlates with the run wall time.

Based on the above analysis, on average, `XtalOpt` consumes 35%, 17%, and 10% of the total CPU time in generating 100, 400, and 1000 structures, respectively. This is an expected result, as the increasing number of generated structures in a search requires more CPU time for local optimizations (and objective calculations, if applicable) performed by the external codes. A breakdown of the CPU time usage for various tasks, averaged over all searches of different settings but the same target structure count, is listed in Table 3. The results show that the overall process of the parent selection is the most demanding task in a search, consuming on average \sim 7% of the code’s CPU usage (0.9% of the total CPU time).

Task	Share of XtalOpt CPU time	Share of total CPU time
Input/Output	1.86%	0.38%
Similarity check	2.53%	0.53%
Fitness analysis and parent selection	6.95%	0.90%
Structure generation and analysis	3.98%	0.77%

Table 3: Relative CPU time usage of various `XtalOpt` tasks, obtained by averaging the results of multiple profiling runs with different input settings; compared to the code and total CPU time usages.

Further, the most computationally-demanding `XtalOpt` functions were identified for various search settings. The average share from the code and total CPU times for each function was calculated over all searches with the same total number of generated structures. According to the average CPU time usages, the non-dominated sorting and comparison of RDF vectors were found to be the most computationally expensive functions in a typical search. Currently, and by default, a non-dominated sorting is conducted regardless

of the global optimization type to obtain and output the Pareto front indices for user's information. This operation, on average, consumes 6% of the overall **XtalOpt** CPU usage (0.8% of the total CPU time) in a typical search, ranking the function as the most computationally expensive task. Second in the CPU usage is, when employed, the comparison of RDF vectors with using 3.7% of the overall **XtalOpt** CPU time (0.7% of the total CPU time).

It should be noted that the share of computational cost of the code and its functions from the total CPU usage of the runs is measured for searches conducted using fast UIP local optimizations. Therefore, the computational overhead of the **XtalOpt** tasks would be insignificant when the structural relaxations are performed using *ab initio* codes.

13.6 Legacy AFLOW-hardness optimization

The explicit support for AFLOW [78] hardness as an objective is removed. AFLOW hardness can still be used as an objective, in the same way as any objective that should be maximized, using an interface executable script. An example of such a script was introduced in the previous release of **XtalOpt** [59].

13.7 Backwards compatibility

Version 14 of **XtalOpt** does not read the input file (`xtalopt.in`) of previous versions due to various changes in the required input flags, and does not resume a run that was performed with an older version of the code. However, old versions of the code can be readily obtained from the **XtalOpt** website [79] and the CPC program library.

14 Conclusions

In the present work, we outlined the latest developments of the **XtalOpt** evolutionary algorithm. The new version of the code expands the multi-objective global optimization (MOGO) feature beyond the generalized scalar fitness function by including the Pareto optimization algorithm. Further, a variable-composition search capability is implemented in the code, which allows the user to choose from various search types, i.e., searching for the ground state of a compound with a fixed composition, searching for (meta)stable phases over a specific set of pre-determined compositions, or exploring the full composition space of the target crystalline chemical system for locating stable phases of variable compositions.

The new release of **XtalOpt** also includes easy-to-use interface scripts for using modern universal interatomic potentials for local optimization of structures. With the described developments, bug fixes, and improvements to the code's workflow, **XtalOpt** version 14 is designed to offer the possibility of an efficient high-throughput exploration of the composition space for a chemical compound to predict novel (meta)stable phases with desired properties.

Acknowledgement

We acknowledge the U.S. National Science Foundation (DMR-2136038 and DMR-2119065) for financial support, and the Center for Computational Research at SUNY Buffalo for computational support (<http://hdl.handle.net/10477/79221>).

Appendix A New implementation in the GUI

The GUI of **XtalOpt** version 14 supports all of the aforementioned developments. Generally, the format of the relevant input entries (e.g., chemical formulas, reference energies, elemental volumes) is the same as in the CLI mode. However, the GUI has been redesigned to accommodate the changes made in this new implementation. The “Structure Limits” and “Search Settings” tabs host the majority of the new settings, while the “Multiobjective Search” tab has a new section (“Optimization”) where the user can choose the optimization type and adjust the relevant settings. Also, the GUI now includes a new “About” tab that offers some basic information and web links where one can learn more about the code. Fig. 4 illustrates a few of the tabs in the new GUI, with the changes relevant to the new implementation highlighted.

New to the GUI of **XtalOpt** is also the option to import run settings from a CLI input file, and to export the GUI parameters to CLI settings file. These options, however, are “experimental” and provided for convenience. The user must verify that the imported parameters in all GUI tabs (or entries in the exported CLI input file) match their expectation.

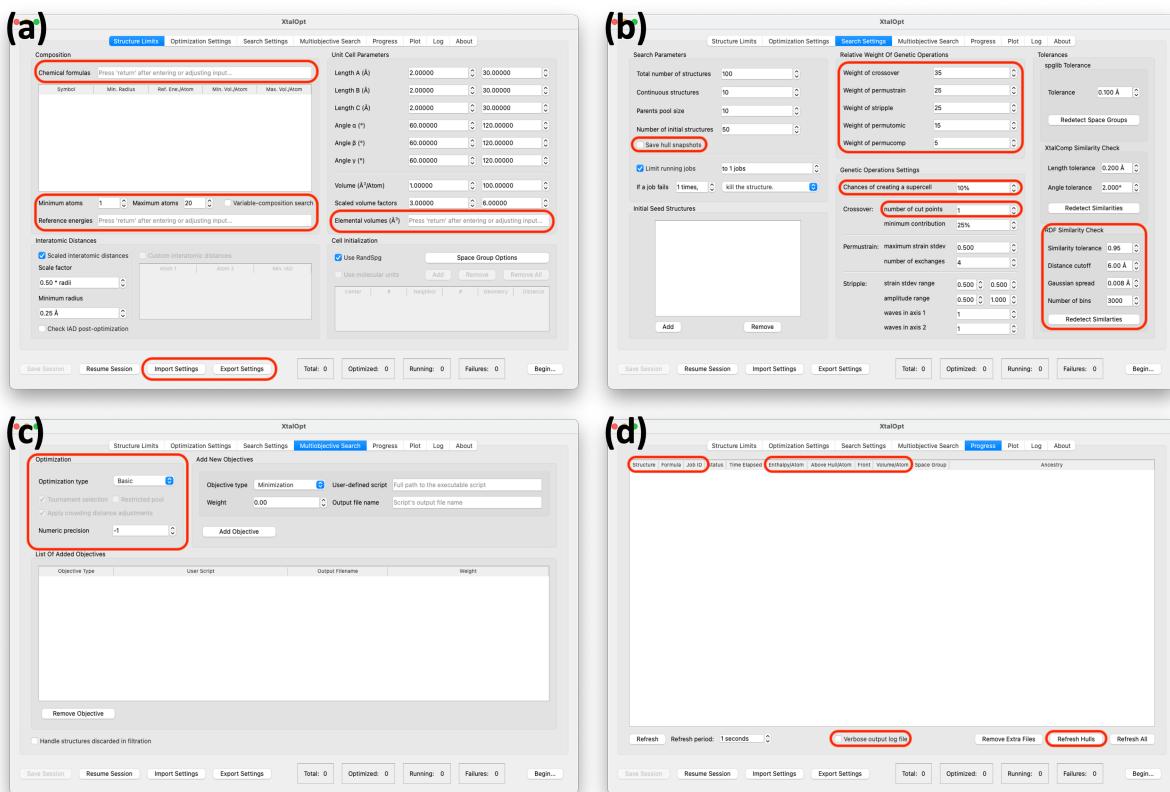


Figure 4: The graphical user interface (GUI) of **XtalOpt** version 14 has been updated to include the new features: (a) the “Structure Limits” tab where the chemical system, search type, reference energies, maximum number of atoms, various unit cell limits and cell initialization schemes for the run can be set; (b) the “Search Settings” tab entries can be used to specify the search duration and run concurrency, action taken when a local optimization fails, initial seed structures, genetic operation specifications, and thresholds for symmetry and similarity detections; (c) the “Multiobjective Search” tab parameters allow for adjusting the global optimization type and its relevant settings, adding objectives for a MO run, and specifying the code’s actions when an external program fails to calculate a filtration objective, and (d) the “Progress” tab maintains a live overview of the produced structures with a summary of their characteristics, such as the chemical formula, enthalpy, distance above the convex hull, Pareto front, symmetry, and ancestry information.

Appendix B Summary of the new, modified, and obsolete CLI flags

New flag	Comments (default values indicated by “*”, if any)	Runtime adjustable
chemicalFormulas	Required input flag: cell formulas of initial generation	No
vcSearch	Logical: perform variable-composition search (*false)	No
minAtoms	Minimum unit cell size generated in the search (*1)	No
maxAtoms	Maximum unit cell size generated in the search (*20)	No
referenceEnergies	Cell formula and energy of convex hull references in local optimizer’s units	No
weightPermutomic	Relative weight of applying permutomic, applicable only to VC search (*15)	Yes
weightPermucomp	Relative weight of applying permucomp, applicable only to VC search (*5)	Yes
crossoverCuts	Number of cut points for crossover operation (*1)	Yes
randomSuperCell	Percent chance of performing random supercell expansion in [0, 100] (*0)	Yes
optimizationType	Optimization scheme: Basic or Pareto (*Basic)	Yes
tournamentSelection	Logical: use tournament selection in Pareto optimization (*true)	Yes
restrictedPool	Logical: restrict the tournament selection to top structures (*false)	Yes
crowdingDistance	Logical: apply crowding distances in Pareto optimization (*true)	Yes
objectivePrecision	Number of decimal digits in objective values for optimization (*-1)	Yes
elementalVolumes	Cell formula and volume limits for elements in Å ³ /cell units	Yes
rdfTolerance	Threshold for RDF dot product to identify similarity in [0.0, 1.0] (*0.0)	Yes
rdfCutoff	Maximum bond length considered for RDF vector in Å (*6.0)	No
rdfSigma	Gaussian spread for RDF calculation in Å (*0.008)	No
rdfNumBins	Number of bins for RDF calculation (*3000)	No
saveHullSnapshots	Logical: save snapshots of hull data (*false)	Yes
verboseOutput	Logical: produce extra information in the run output (*false)	Yes
user1	Custom user-defined keyword	No
user2	Custom user-defined keyword	No
user3	Custom user-defined keyword	No
user4	Custom user-defined keyword	No

Old flag	Renamed/changed flag	Comments (default values indicated by “*”, if any)	Runtime adjustable
popSize	parentsPoolSize		Yes
volumeMin	minVolume	Value should be in Å ³ per atom	Yes
volumeMax	maxVolume	Value should be in Å ³ per atom	Yes
volumeScaleMin	minVolumeScale	Factor of “covalent” sphere	Yes
volumeScaleMax	maxVolumeScale	Factor of “covalent” sphere	Yes
percentChanceCrossover	weightCrossover	Relative weight of applying operation (*35)	Yes
percentChanceStripple	weightStripple	Relative weight of applying operation (*25)	Yes
percentChancePermustrain	weightPermustrain	Relative weight of applying operation (*25)	Yes
objective		All entries are necessary	No
seedStructures		A comma-delimited list	No

Obsolete flag(s)	Comments
empiricalFormula	Replaced with “chemicalFormulas” flag
formulaUnits	Merged into “chemicalFormulas” flag
usingFormulaUnitCrossovers	It is performed by default
usingOneGenePool	It is performed by default
mitosisA, mitosisB, mitosisC, printSubcell, mitosisDivisions, chanceOffFutureMitosis, usingMitoticGrowth, usingSubcellMitosis,	Mitosis feature and the corresponding flags are removed in XtalOpt version 14. See the “random supercell expansion” (and “randomSuperCell” flag) for a substitute functionality.

Appendix C Example input files for an XtalOpt run

XtalOpt input file for a VC search for the Ti-O system, using the MACE potential.

The content of the `xtalopt.in` file

```
softExit = true
chemicalFormulas = Ti7 01, Ti1 07
referenceEnergies = Ti3 -23.35901499 , 016 -78.97000885
vcSearch = true
maxAtoms = 40
numInitial = 50
parentsPoolSize = 20
limitRunningJobs = true
runningJobLimit = 2
continuousStructures = 15
maxNumStructures = 500
usingRandSpg = true
minVolumeScale = 3.0
maxVolumeScale = 6.0
weightCrossover = 35
weightStripple = 25
weightPermustrain = 25
weightPermutomic = 15
weightPermucomp = 5
randomSuperCell = 10
radiiScalingFactor = 0.5
aMin = 2.0
bMin = 2.0
cMin = 2.0
aMax = 30.0
bMax = 30.0
cMax = 30.0
alphaMin = 60.0
betaMin = 60.0
gammaMin = 60.0
alphaMax = 120.0
betaMax = 120.0
gammaMax = 120.0
templatesDirectory = ./templates
queueInterface = local
localWorkingDirectory = ./local
numOptimizationSteps = 1
optimizer = vasp
exeLocation = /Users/sam/TiO/vasp_uip.sh
incarTemplates = incar1
kpointsTemplates = kpoints1
potcarFile system = ./templates/potcar1
jobFailLimit = 1
jobFailAction = kill
rdfTolerance = 0.95
spglibTolerance = 0.01
```

Bash executable script to run the UIP interface script; to be used as a “VASP”-like optimizer for `XtalOpt` run.

The content of `vasp_uip.sh` script

```
#!/bin/bash

# Path to the UIP interface script
exe=/usr/local/bin/vasp_uip.py

# Load python environment
source /Users/sam/penv/bin/activate

# Relaxation task
python $exe -n 75 -o
```

With the above files, and with a `templates/` folder that contains (empty) files `incar1`, `kpoints1`, and `potcar1`, one can run `XtalOpt` as:

```
/path/to/xtalopt --cli
```

After the run is finished, the `pycxl` script can be used to plot the convex hull of the run, e.g.:

```
python /path/to/pycxl.py -i local/hull.txt -y -4 1 -c 1
```

This produces the `out_distances.pdf` file, for which an example output is illustrated in Figure 5.

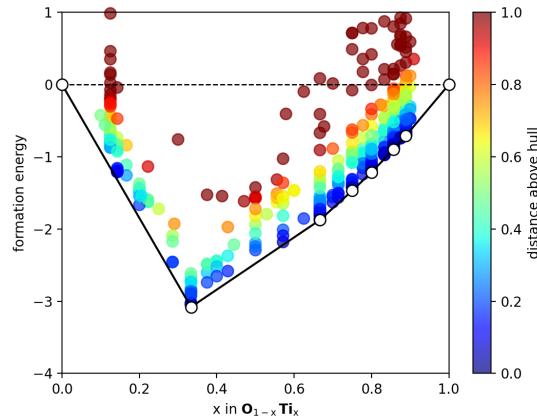


Figure 5: An example of the convex hull produced by the `pycxl` code, from the output of the `XtalOpt` VC run for the Ti-O system.

References

- [1] M. Alverson, S. G. Baird, R. Murdock, S. H. Ho, J. Johnson, et al., Generative adversarial networks and diffusion models in material discovery, *Digital Discovery* 3 (1) (2023) 62–80. doi:[10.1039/d3dd00137g](https://doi.org/10.1039/d3dd00137g).
- [2] X. Luo, Z. Wang, P. Gao, J. Lv, Y. Wang, et al., Deep learning generative model for crystal structure prediction, *npj Computational Materials* 10 (1) (2024). doi:[10.1038/s41524-024-01443-y](https://doi.org/10.1038/s41524-024-01443-y).
- [3] S. Yang, K. H. Cho, A. Merchant, P. Abbeel, D. Schuurmans, et al., Scalable Diffusion for Materials Generation, 12th International Conference on Learning Representations, ICLR 2024 (2024). doi:[10.48550/ARXIV.2311.09235](https://doi.org/10.48550/ARXIV.2311.09235).
- [4] C. Zeni, R. Pinsler, D. Züigner, A. Fowler, M. Horton, et al., A generative model for inorganic materials design, *Nature* 639 (8055) (2025) 624–632. doi:[10.1038/s41586-025-08628-5](https://doi.org/10.1038/s41586-025-08628-5).
- [5] Y.-W. Zhang, V. Sorkin, Z. H. Aitken, A. Politano, J. Behler, et al., Roadmap for the development of machine learning-based interatomic potentials, *Modelling and Simulation in Materials Science and Engineering* 33 (2) (2025) 023301. doi:[10.1088/1361-651X/ad9d63](https://doi.org/10.1088/1361-651X/ad9d63).
- [6] V. Stanev, C. Oses, A. G. Kusne, E. Rodriguez, J. Paglione, et al., Machine learning modeling of superconducting critical temperature, *npj Computational Materials* 4 (1) (2018) 29. doi:[10.1038/s41524-018-0085-8](https://doi.org/10.1038/s41524-018-0085-8).
- [7] Y. Zhuo, A. Mansouri Tehrani, J. Brgoch, Predicting the Band Gaps of Inorganic Solids by Machine Learning, *Journal of Physical Chemistry Letters* 9 (7) (2018) 1668–1673. doi:[10.1021/acs.jpclett.8b00124](https://doi.org/10.1021/acs.jpclett.8b00124).
- [8] R. E. Goodall, A. A. Lee, Predicting materials properties without crystal structure: deep representation learning from stoichiometry, *Nature Communications* 11 (1) (2020) 6280. doi:[10.1038/s41467-020-19964-7](https://doi.org/10.1038/s41467-020-19964-7).
- [9] C. J. Court, J. M. Cole, Magnetic and superconducting phase diagrams and transition temperatures predicted using text mining and machine learning, *npj Computational Materials* 6 (1) (2020) 18. doi:[10.1038/s41524-020-0287-8](https://doi.org/10.1038/s41524-020-0287-8).
- [10] G. L. Hart, T. Mueller, C. Toher, S. Curtarolo, Machine learning for alloys, *Nature Reviews Materials* 6 (8) (2021) 730–755. doi:[10.1038/s41578-021-00340-w](https://doi.org/10.1038/s41578-021-00340-w).
- [11] J. S. Kim, J. Noh, J. Im, Machine learning-enabled chemical space exploration of all-inorganic perovskites for photovoltaics, *npj Computational Materials* 10 (1) (2024) 97. doi:[10.1038/s41524-024-01270-1](https://doi.org/10.1038/s41524-024-01270-1).
- [12] Z. Wang, X. Luo, Q. Wang, H. Ge, P. Gao, et al., Advances in high-pressure materials discovery enabled by machine learning, *Matter and Radiation at Extremes* 10 (3) (2025) 033801. doi:[10.1063/5.0255385](https://doi.org/10.1063/5.0255385).
- [13] J. Gan, P. Zhong, Y. Du, Y. Zhu, C. Duan, et al., Large Language Models Are Innate Crystal Structure Generators, *arXiv* (2025). doi:[10.48550/arXiv.2502.20933](https://doi.org/10.48550/arXiv.2502.20933).
- [14] E. Zurek, Discovering New Materials via A Priori Crystal Structure Prediction, *Reviews in Computational Chemistry* 29 (2016) 274–326. doi:[10.1002/9781119148739.ch5](https://doi.org/10.1002/9781119148739.ch5).
- [15] A. R. Oganov, C. J. Pickard, Q. Zhu, R. J. Needs, Structure prediction drives materials discovery, *Nature Reviews Materials* 4 (5) (2019) 331–348. doi:[10.1038/s41578-019-0101-8](https://doi.org/10.1038/s41578-019-0101-8).
- [16] Z. Falls, P. Avery, X. Wang, K. P. Hilleke, E. Zurek, The XtalOpt Evolutionary Algorithm for Crystal Structure Prediction, *Journal of Physical Chemistry C* 125 (3) (2021) 1601–1620. doi:[10.1021/acs.jpcc.0c09531](https://doi.org/10.1021/acs.jpcc.0c09531).
- [17] Y. Wang, J. Lv, P. Gao, Y. Ma, Crystal Structure Prediction via Efficient Sampling of the Potential Energy Surface, *Accounts of Chemical Research* 55 (15) (2022) 2068–2076. doi:[10.1021/acs.accounts.2c00243](https://doi.org/10.1021/acs.accounts.2c00243).

- [18] L. J. Conway, C. J. Pickard, A. Hermann, First principles crystal structure prediction, in: Comprehensive Inorganic Chemistry III, Third Edition, Vol. 1-10, Elsevier, 2023, pp. 393–420. doi:10.1016/B978-0-12-823144-9.00173-4.
- [19] W. Ibarra-Hernández, S. Hajinazar, G. Avendaño-Franco, A. Bautista-Hernández, A. N. Kolmogorov, et al., Structural search for stable Mg-Ca alloys accelerated with a neural network interatomic model, *Physical Chemistry Chemical Physics* 20 (43) (2018) 27545–27557. doi:10.1039/c8cp05314f.
- [20] S. Hajinazar, E. D. Sandoval, A. J. Cullo, A. N. Kolmogorov, Multitribe evolutionary search for stable Cu-Pd-Ag nanoparticles using neural network models, *Physical Chemistry Chemical Physics* 21 (17) (2019) 8729–8742. doi:10.1039/c9cp00837c.
- [21] A. Thorn, J. Rojas-Nunez, S. Hajinazar, S. E. Baltazar, A. N. Kolmogorov, Toward ab Initio Ground States of Gold Clusters via Neural Network Modeling, *Journal of Physical Chemistry C* 123 (50) (2019) 30088–30098. doi:10.1021/acs.jpcc.9b08517.
- [22] S. Kharabadze, A. Thorn, E. A. Koulakova, A. N. Kolmogorov, Prediction of stable Li-Sn compounds: boosting ab initio searches with neural network potentials, *npj Computational Materials* 8 (1) (2022) 136. doi:10.1038/s41524-022-00825-4.
- [23] X. Wang, Z. Wang, P. Gao, C. Zhang, J. Lv, et al., Data-driven prediction of complex crystal structures of dense lithium, *Nature Communications* 14 (1) (2023) 2924. doi:10.1038/s41467-023-38650-y.
- [24] P. T. Salzbrenner, S. H. Joo, L. J. Conway, P. I. Cooke, B. Zhu, et al., Developments and further applications of ephemeral data derived potentials, *Journal of Chemical Physics* 159 (14) (2023) 144801. doi:10.1063/5.0158710.
- [25] J. Roberts, B. Rijal, S. Divilov, J. P. Maria, W. G. Fahrenholtz, et al., Machine learned interatomic potentials for ternary carbides trained on the AFLOW database, *npj Computational Materials* 10 (1) (2024). doi:10.1038/s41524-024-01321-7.
- [26] D. C. Lonie, E. Zurek, XtalOpt: An open-source evolutionary algorithm for crystal structure prediction, *Computer Physics Communications* 182 (2) (2011) 372–387. doi:10.1016/j.cpc.2010.07.048.
- [27] S. Hajinazar, A. Thorn, E. D. Sandoval, S. Kharabadze, A. N. Kolmogorov, MAISE: Construction of neural network interatomic models and evolutionary structure optimization, *Computer Physics Communications* 259 (2021) 107679. doi:10.1016/j.cpc.2020.107679.
- [28] W. W. Tipton, R. G. Hennig, A grand canonical genetic algorithm for the prediction of multi-component phase diagrams and testing of empirical potentials, *Journal of Physics Condensed Matter* 25 (49) (2013) 495401. doi:10.1088/0953-8984/25/49/495401.
- [29] A. R. Oganov, C. W. Glass, Crystal structure prediction using ab initio evolutionary techniques: Principles and applications, *Journal of Chemical Physics* 124 (24) (2006) 244704. doi:10.1063/1.2210932.
- [30] Y. Wang, J. Lv, L. Zhu, Y. Ma, Crystal structure prediction via particle-swarm optimization, *Physical Review B - Condensed Matter and Materials Physics* 82 (9) (2010) 94116. doi:10.1103/PhysRevB.82.094116.
- [31] C. J. Pickard, R. J. Needs, Structures at high pressure from random searching, *Physica Status Solidi (B) Basic Research* 246 (3) (2009) 536–540. doi:10.1002/pssb.200880546.
- [32] I. Giagkiozis, P. J. Fleming, Methods for multi-objective optimization: An analysis, *Information Sciences* 293 (2015) 338–350. doi:10.1016/j.ins.2014.08.071.
- [33] N. Gunantara, A review of multi-objective optimization: Methods and its applications, *Cogent Engineering* 5 (1) (2018) 1–16. doi:10.1080/23311916.2018.1502242.
- [34] A. Solomou, G. Zhao, S. Boluki, J. K. Joy, X. Qian, et al., Multi-objective Bayesian materials discovery: Application on the discovery of precipitation strengthened NiTi shape memory alloys through micromechanical modeling, *Materials and Design* 160 (2018) 810–827. doi:10.1016/j.matdes.2018.10.014.

- [35] A. M. Gopakumar, P. V. Balachandran, D. Xue, J. E. Gubernatis, T. Lookman, Multi-objective Optimization for Materials Discovery via Adaptive Design, *Scientific Reports* 8 (1) (2018) 3738. doi:10.1038/s41598-018-21936-3.
- [36] D. Khatamsaz, B. Vela, P. Singh, D. D. Johnson, D. Allaire, et al., Multi-objective materials bayesian optimization with active learning of design constraints: Design of ductile refractory multi-principal-element alloys, *Acta Materialia* 236 (2022) 118133. doi:10.1016/j.actamat.2022.118133.
- [37] H. Z. Chen, Y. Y. Zhang, X. Gong, H. Xiang, Predicting new TiO₂ phases with low band gaps by a multiobjective global optimization approach, *Journal of Physical Chemistry C* 118 (5) (2014) 2333–2337. doi:10.1021/jp411437f.
- [38] Y. Y. Zhang, W. Gao, S. Chen, H. Xiang, X. G. Gong, Inverse design of materials by multi-objective differential evolution, *Computational Materials Science* 98 (1) (2015) 51–55. doi:10.1016/j.commatsci.2014.10.054.
- [39] M. Núñez-Valdez, Z. Allahyari, T. Fan, A. R. Oganov, Efficient technique for computational design of thermoelectric materials, *Computer Physics Communications* 222 (2018) 152–157. doi:10.1016/j.cpc.2017.10.001.
- [40] J. J. Maldonis, Z. Xu, Z. Song, M. Yu, T. Mayeshiba, et al., StructOpt: A modular materials structure optimization suite incorporating experimental data and simulated energies, *Computational Materials Science* 160 (2019) 1–8. doi:10.1016/j.commatsci.2018.12.052.
- [41] C. Y. Cheng, J. E. Campbell, G. M. Day, Evolutionary chemical space exploration for functional materials: computational organic semiconductor discovery, *Chemical Science* 11 (19) (2020) 4922–4933. doi:10.1039/d0sc00554a.
- [42] J. Meng, M. Abbasi, Y. Dong, C. Carlos, X. Wang, et al., Experimentally informed structure optimization of amorphous TiO₂ films grown by atomic layer deposition, *Nanoscale* 15 (2) (2022) 718–729. doi:10.1039/d2nr03614b.
- [43] S. Racioppi, A. Otero-De-la Roza, S. Hajinazar, E. Zurek, Powder X-ray diffraction assisted evolutionary algorithm for crystal structure prediction, *Digital Discovery* 4 (1) (2025) 73–83. doi:10.1039/d4dd00269e.
- [44] G. Trimarchi, A. J. Freeman, A. Zunger, Predicting stable stoichiometries of compounds via evolutionary global space-group optimization, *Physical Review B - Condensed Matter and Materials Physics* 80 (9) (2009) 92101. doi:10.1103/PhysRevB.80.092101.
- [45] X. Zhang, A. Zunger, G. Trimarchi, Structure prediction and targeted synthesis: A new NanN₂ diazenide crystalline structure, *The Journal of Chemical Physics* 133 (19) (2010) 194504. doi:10.1063/1.3488440.
- [46] W. W. Tipton, C. R. Bealing, K. Mathew, R. G. Hennig, Structures, phase stabilities, and electrical potentials of Li-Si battery anode materials, *Physical Review B - Condensed Matter and Materials Physics* 87 (18) (2013) 184114. doi:10.1103/PhysRevB.87.184114.
- [47] B. C. Revard, W. W. Tipton, A. Yesypenko, R. G. Hennig, Grand-canonical evolutionary algorithm for the prediction of two-dimensional materials, *Physical Review B* 93 (5) (2016) 54117. doi:10.1103/PhysRevB.93.054117.
- [48] X. Yang, H. Li, H. Liu, H. Wang, Y. Yao, et al., Pressure-induced decomposition of binary lanthanum intermetallic compounds, *Phys. Rev. B* 101 (2020) 184113. doi:10.1103/PhysRevB.101.184113.
- [49] Q. Zhu, D. Y. Jung, A. R. Oganov, C. W. Glass, C. Gatti, et al., Stability of xenon oxides at high pressures, *Nature Chemistry* 5 (1) (2013) 61–65. doi:10.1038/nchem.1497.
- [50] Y. Liu, A. R. Oganov, S. Wang, Q. Zhu, X. Dong, et al., Prediction of new thermodynamically stable aluminum oxides, *Scientific Reports* 5 (1) (2015) 9518. doi:10.1038/srep09518.

- [51] A. G. Kvashnin, C. Tantardini, H. A. Zakaryan, Y. A. Kvashnina, A. R. Oganov, Computational Search for New W–Mo–B Compounds, *Chemistry of Materials* 32 (16) (2020) 7028–7035. doi:10.1021/acs.chemmater.0c02440.
- [52] I. Batatia, P. Benner, Y. Chiang, A. M. Elena, D. P. Kovács, et al., A foundation model for atomistic materials chemistry, arXiv (2023). doi:10.48550/arxiv.2401.00096.
- [53] B. Deng, P. Zhong, K. J. Jun, J. Riebesell, K. Han, et al., CHGNet as a pretrained universal neural network potential for charge-informed atomistic modelling, *Nature Machine Intelligence* 5 (9) (2023) 1031–1041. doi:10.1038/s42256-023-00716-3.
- [54] B. Rhodes, S. Vandenhaute, V. Šimkus, J. Gin, J. Godwin, et al., Orb-v3: atomistic simulation at scale, arXiv (2025). doi:10.48550/arXiv.2504.06231.
- [55] H. Yang, C. Hu, Y. Zhou, X. Liu, Y. Shi, et al., MatterSim: A Deep Learning Atomistic Model Across Elements, Temperatures and Pressures, arXiv (2024). doi:10.48550/arXiv.2405.04967.
- [56] C. Chen, S. P. Ong, A universal graph deep learning interatomic potential for the periodic table, *Nature Computational Science* 2 (11) (2022) 718–728. doi:10.1038/s43588-022-00349-3.
- [57] K. Choudhary, B. De Cost, L. Major, K. Butler, J. Thiyyagalingam, et al., Unified graph neural network force-field for the periodic table: solid state applications, *Digital Discovery* 2 (2) (2023) 346–355. doi:10.1039/d2dd00096b.
- [58] A. Merchant, S. Batzner, S. S. Schoenholz, M. Aykol, G. Cheon, et al., Scaling deep learning for materials discovery, *Nature* 624 (7990) (2023) 80–85. doi:10.1038/s41586-023-06735-9.
- [59] S. Hajinazar, E. Zurek, XtalOpt version 13: Multi-objective evolutionary search for novel functional materials, *Computer Physics Communications* 304 (2024) 109306. doi:10.1016/j.cpc.2024.109306.
- [60] D. C. Lonie, E. Zurek, XtalOpt version r7: An open-source evolutionary algorithm for crystal structure prediction, *Computer Physics Communications* 182 (10) (2011) 2305–2306. doi:10.1016/j.cpc.2011.06.003.
- [61] Z. Falls, D. C. Lonie, P. Avery, A. Shamp, E. Zurek, XTALOPT version r9: An open-source evolutionary algorithm for crystal structure prediction, *Computer Physics Communications* 199 (2016) 178–179. doi:10.1016/j.cpc.2015.09.018.
- [62] P. Avery, C. Toher, S. Curtarolo, E. Zurek, XTALOPT Version r12: An open-source evolutionary algorithm for crystal structure prediction, *Computer Physics Communications* 237 (2019) 274–275. doi:10.1016/j.cpc.2018.11.016.
- [63] P. Avery, X. Wang, C. Osse, E. Gossett, D. M. Proserpio, et al., Predicting superhard materials via a machine learning informed evolutionary structure search, *npj Computational Materials* 5 (1) (2019) 89. doi:10.1038/s41524-019-0226-8.
- [64] X. Wang, D. M. Proserpio, C. Osse, C. Toher, S. Curtarolo, et al., The Microscopic Diamond Anvil Cell: Stabilization of Superhard, Superconducting Carbon Allotropes at Ambient Pressure, *Angewandte Chemie - International Edition* 61 (32) (2022) e202205129. doi:10.1002/anie.202205129.
- [65] B. Wang, K. P. Hilleke, S. Hajinazar, G. Frapper, E. Zurek, Structurally Constrained Evolutionary Algorithm for the Discovery and Design of Metastable Phases, *Journal of Chemical Theory and Computation* 19 (21) (2023) 7960–7971. doi:10.1021/acs.jctc.3c00594.
- [66] P. Avery, E. Zurek, RandSpg: An open-source program for generating atomistic crystal structures with specific spacegroups, *Computer Physics Communications* 213 (2017) 208–216. doi:10.1016/j.cpc.2016.12.005.
- [67] C. B. Barber, D. P. Dobkin, H. Huhdanpaa, The Quickhull Algorithm for Convex Hulls, *ACM Transactions on Mathematical Software* 22 (4) (1996) 469–483. doi:10.1145/235815.235821.

- [68] S. Racioppi, E. Zurek, Using Topology to Predict Electrides in the Solid State, arXiv (2025). doi: 10.48550/arXiv.2508.04548.
- [69] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197. doi:10.1109/4235.996017.
- [70] D. C. Lonie, E. Zurek, Identifying duplicate crystal structures: XtalComp, an open-source solution, Computer Physics Communications 183 (3) (2012) 690–697. doi:10.1016/j.cpc.2011.11.007.
- [71] E. Zurek, Y. Yao, Theoretical predictions of novel superconducting phases of BaGe₃ stable at atmospheric and high pressures, Inorganic Chemistry 54 (6) (2015) 2875–2884. doi:10.1021/ic5030235.
- [72] E. L. Willighagen, R. Wehrens, P. Verwer, R. de Gelder, L. M. C. Buydens, Method for the computational comparison of crystal structures, Acta Crystallographica Section B 61 (1) (2005) 29–36. doi:10.1107/S0108768104028344.
- [73] J. A. Chisholm, S. Motherwell, COMPACK: a program for identifying crystal structure similarity using distances, Journal of Applied Crystallography 38 (1) (2005) 228–231. doi:10.1107/S0021889804027074.
- [74] N. L. Abraham, M. I. Probert, Improved real-space genetic algorithm for crystal structure and polymorph prediction, Physical Review B - Condensed Matter and Materials Physics 77 (13) (2008) 134117. doi:10.1103/PhysRevB.77.134117.
- [75] P. Avery, Z. Falls, E. Zurek, XTALOPT Version r10: An open-source evolutionary algorithm for crystal structure prediction, Computer Physics Communications 217 (2017) 210–211. doi:10.1016/j.cpc.2017.04.001.
- [76] A. Hjorth Larsen, J. JØrgen Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, et al., The atomic simulation environment - A Python library for working with atoms, Journal of Physics Condensed Matter 29 (27) (2017) 273002. doi:10.1088/1361-648X/aa680e.
- [77] S. Hajinazar, pycxl: Python Script to Calculate Convex Hull, <https://github.com/hajinazar/pycxl>.
- [78] S. Curtarolo, W. Setyawan, G. L. Hart, M. Jahnatek, R. V. Chepulskii, et al., AFLOW: An automatic framework for high-throughput materials discovery, Computational Materials Science 58 (2012) 218–226. doi:10.1016/j.commatsci.2012.02.005.
- [79] XtalOpt Website, <https://xtalopt.github.io>, accessed: 2025-07-10.