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

1 Introduction

Previous versions of `XtalOpt` [1, 2], available in both command-line interface (CLI) and graphical user interface (GUI) modes, performed evolutionary search on a chemical system specified by its reduced empirical formula, e.g.,

```
empiricalFormula = Ti1O2
```

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

```
formulaUnits = 1-4
```

This resulted in a search over a "fixed chemical composition". In this search, the initial set of structures are generated from the specified supercells of the given empirical formula. Then, the new structures are produced with the size and composition matching that of the initial list, by applying genetic operations to the selected structures from the parent pool.

In this workflow, (i) the selection of the parent structure(s) is based on the fitness of structure, (ii) the fitness of a structure is determined from the normalized relative enthalpies (for a single-objective search) and -possibly- normalized relative value of user-defined objectives (in a multi-objective) search, (ii) the genetic operations include: "crossover" where two parent structure are mixed to generate an offspring with the same composition, and "stripple" and "permustrain" mutations that generate a new structure by applying random distortions to a selected parent structure.

In the new version of `XtalOpt` (version 14), briefly,

- Besides the generalized fitness function, the "Pareto optimization" scheme is implemented,
- Instead of directly using the enthalpy to measure the structure fitness, the "distance above the convex hull" is used,

- The genetic operation "crossover" is generalized and new evolutionary operations are introduced to allow for the generation of structures with new stoichiometries, and
- It is possible to conduct the search over multiple compositions, or to generate new structures with compositions that are not listed in the input set of compositions.

2 Input chemical formulas

In the new version of `XtalOpt`, introducing the chemical system and -optionally- desired formula units is different than the previous versions: (i) the "empiricalFormula" and "formulaUnits" pair of flags are 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., combination of element symbols and corresponding atom counts, e.g., "Ti1O4" instead of "TiO4").

That is to say that the input:

```
chemicalFormulas = Ti4O8
```

instructs `XtalOpt` to search in 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 the "formula2" is a proper supercell of the "formula1". That's, the number of atoms of each element type in "formula2" are 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
```

For comparison, the above input is corresponding to specifying the pair of "empiricalFormula = Ti1O2" and "formulaUnits = 1-4, 8" input flags in the previous versions of code.

The "chemicalFormulas" parameter not only determine the initial cell(s) to be generated 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: initial set of structures are 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 and outcome of the genetic operations are different. That's, 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 follows.

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, Ti1 O1 - Ti5O5
```

This input instructs `XtalOpt` to perform a "multi-composition" (MC) search. This type of search is similar to the FC search in that initial set of structures are generated as usual and subsequent cells generated in the search are forced to have a composition matching that of "one of their parent structure(s)". 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 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, genetic operation crossover is now allowed to produce cells with new compositions that are not necessarily match their parent structures' composition or even those listed among the initial chemical formulas. This is called a "variable-composition" (VC) search: the most general type of evolutionary search conducted by `XtalOpt` where the entire chemical system of the specified elements is explored to find (meta)stable phases of various compositions.

4 Maximum number of atoms

The new version of code allows for generating structures of new composition (hence, various number of atoms in the cell) and random supercells (discussed below). In order to control the computational cost of the search, a new input parameter, "maxAtoms", is introduced that sets the maximum number of atoms in the generated unit cells during the search.

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 the set limit.

The default value for 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. The user can also modify this at the run time.

5 Genetic operations

`XtalOpt` uses various genetic operations to produce new structures from selected parent structure(s). In previous versions, the defined operations included crossover, stripple, and permustrain. 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, must sum to 100.

In the new version of the code, 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 for chances of choosing genetic operations and the interpretation of their values are changed, as follows.

5.1 New genetic operation: permutomic

Applicable only to a VC search, a new evolutionary operation, "permutomic", is introduced that randomly adds (removes) an atom to (from) a structure chosen from the parent pool. For such a search, the user should provide a probability for performing this operation in the input, which is adjustable during the run.

5.2 New genetic operation: permucomp

Also applicable only to a VC search, the new genetic operation "permucomp" is introduced that randomly creates a random new composition with a randomly chosen total number of atoms (up to "maxAtoms" parameter).

It should be noted that this random mutation is primarily designed to diversity the structures pool in long searches over multiple elements. With a limited chance of resulting in an energetically favorable structure, ideally it should be assigned a relatively small probability compared to other genetic operations.

5.3 Chances of performing the genetic operations

In the new version of `XtalOpt`, instead of "percent chance" for applying genetic operations, the probability of choosing the genetic operations should be specified by a "relative weight".

In the CLI input, and instead of using "percentChances..." flags, this can be done using the set of flags:

```
weightCrossover
weightStripple
weightPermustrain
weightPermutomic
weightPermucomp
```

Weight for each operation should be zero or a positive integer (arbitrary otherwise) that can also be modified during the run, with no specific condition for their total value.

During the search and at the time of applying genetic operations, 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, from:

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

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 considered zero.

Input Flag	Default value	Percent chance (VC)	Percent chance (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 weight of various genetic operations, with their default values. The actual percentage chances of applying a genetic operation, depending on the search type, is listed in the last columns.

The default values for relative operation weights, and the corresponding runtime -approximate- percent chance for various search types are summarized in the 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 proabibility for applying the genetic operations while a warning message is printed in the run output.

5.4 Random mutation: supercell generation

Although not a genetic operation per se, user can optionally define a finite probability as a precent chances (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 by a non-zero value, the expansion factor will be chosen by `XtalOpt` randomly, as a set of three integers, such that the 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.

6 Reference energies

As of version 14, `XtalOpt` uses "distance above convex hull" as the target value for energy optimization. The calculation of the convex hull (performed using the `Qhull` library [3]), requires defining reference energies. By default, the code uses "0.0" as the "elemental reference energies". Correct convex hull for MC and VC searches, however, needs correct reference energies to be specified.

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, e.g., for Ti-O system:

```
referenceEnergies = O1 -2.53, Ti4 -17.65
```

The energies introduced in the input must have the same units as is being used by the local optimizer, in order to produce the correct results.

Further, for any system, the user can introduce reference energies of not only elements but for relevant multi-element (sub)systems; if they are needed for calculating the correct convex hull.

For example, for O-Ti-N system, this can be a valid entry:

```
referenceEnergies = O2 -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 the previous release of `XtalOpt`, the multi-objective optimization feature was implemented to facilitate the search for functional materials by introducing a basic generalized fitness function as the weighted sum of normalized values of objectives. Regardless of the number of objectives, by default, `XtalOpt` employs this scalar fitness function for global optimization (i.e., "Basic" optimization). However, the new version of the code is further extended by including the *Pareto optimization* as an additional optimization scheme, in principle applicable to both single- and multi-objective searches. The user can instruct the code to utilize the Pareto optimization through the input flag:

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

For Pareto optimization, `XtalOpt` uses the NSGA-II algorithm [4] in which the Pareto front (rank) and crowding distances of the structures in the parents 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 from the selected pair as the one that has a better rank, or a greater crowding distance if both belong to the same Pareto front. For structures of similar rank and crowding distance, the selection is made randomly.

It should be noted that 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; unlike the scalar fitness (basic) scheme where a limited subset of "top" candidate structures (i.e., equal to the parent pool size 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 crowding distances within each rank), and select the "top" candidates such that there will be as many as the user-specified parent pool size structure from which the tournament selection is performed.

Besides the tournament selection, `XtalOpt` provides another parent selection option within the Pareto optimization. That's, to use the obtained rank and crowding distances of the structures to calculate a scalar fitness, and make the parent selection based on this measure from the top candidate structures, 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 structure, 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

$$f_r^0 = \frac{N - r}{N} \tag{1}$$

that is corresponding 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:

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

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 the 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 based on structure ranks, 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 especially the outcome of non-dominated sorting is sensistive to the numerical precision of the objective values. Although `XtalOpt` does not manipulate the calculated values of objectives, it provides the user with the option to apply a desired precision to the objective values, prior to parent selection process. A user-specified number of decimal digits will be retained for objective values as determined using the flag:

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

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

For a multi-objective search in the previous version of `XtalOpt`, specifying the objective weight and output file name for the executable script was optional in the CLI mode. In the new version of code, these fields are all mandatory. That's, all four input entries for an objective must be given in the below order:

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

Further, 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 be always specified since the user has the option to change the optimization type at the runtime ("Pareto" to "Basic, and vice versa), and
- Just similar to the previous workflow, any minimizable/maximizable objective with the weight of zero, will be calculated but will not be considered in the scalar fitness for the basic optimization.

8 Volume limits

In the previous versions of `XtalOpt`, the volume of the unit cell could be constrained through either absolute minimum and maximum limits (in Å^3 per formula unit) or the scaled volume limits.

In the new version of the code, the absolute volume limits are read in the "Å^3 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
```

It should be noted that, however, to be consist with minimum atomic radii settings, the scaled volume limits in the new version of `XtalOpt` are considered as factors of spheres with "covalent" radii of atoms (unlike the previous version, in which the spheres of van der Waals radius were used to obtain the volume limits). As a result, conversion from old scaled volume input values to the new ones would require a factor of $\sim$1.5.

Moreover, in the new version of `XtalOpt`, the user has the option to define volume limits for elements. This setting will take effect only if the limits are defined for all elements in the chemical system, which can be done through specifying a list of comma-delimited entries. Each entry includes a full chemical formula for the elemental unit cell followed by the corresponding volume limits (minimum and maximum, respectively) in Å^3 units, e.g.,

```
elementalVolumes = O1 20 40 , Ti2 50 100
```

Generally, the volume constrains are used in `XtalOpt` 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 Å^3 per atom) are used.

9 Similarity check

In the previous versions of the code the structures were examined after optimization to detect similar structures and exclude them from the parents pool to prevent the search from being biased by a specific structural motif. This process was called "duplicate check" and such structures were labeled as "duplicate" in the output files. Checking the population for detecting the type of structures were done using the `XtalComp` library [5]. In the new version, there are changes regarding this process, as follows.

9.1 Similarity or duplicate check?

Since the output of the `XtalComp` analysis depends on the specified tolerances, two structures which are marked as duplicate might not be an exact match structure-wise. In fact, for the purpose of evolutionary search, the important criterion for curating the parents pool is the "close similarity" of structures, and not necessarily an exact match. In the new version of the code the name for this process is changed to "similarity check", and the structures detected in this process 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 similarity check.

9.2 A new similarity check option

As of `XtalOpt` version 14, besides the `XtalComp`, the user can instruct the code to use the radial distribution function (RDF) of species-resolved bonds to detect similarities between structures. The implemented methodology is similar to that of introduced in the `MAISE` code [6].

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 RDF similarity check instead of `XtalComp`. That's, it calculates the scalar product of normalized RDF vector of structures, and those which 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 being used for similarity check.

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

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

These settings can be specified and adjusted during the run by the user. 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 the previous versions of `XtalOpt`, the seed structures, with the same composition of the reference chemical system of the search, could be introduced using a space-separated list. Now, for consistency, this should be a comma-delimited list of entries, e.g.,

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

Moreover, the new version of `XtalOpt` allows for introducing seed structures that are a "sub-system" of the reference chemical system, or have a composition that is not listed in the input formulas list. That's, e.g., adding binary or elemental structures in a search for a ternary system. These "off-composition" seeds will be read in as long as they do not include any element which is not in the reference chemical system.

The off-composition seed structures will be included in the parents pool upon successful local optimization, and will participate in the genetic operations if selected as a parent structure depending on their fitness value. This particular feature is useful as instead of specifying reference energies, the user has the option to directly introduce "reference structures" as seed. This helps to ensure that the obtained reference energies are consistent with the local optimization settings of the search. This option, however, should be utilized carefully since if the external code fails to optimize the reference structures, calculated convex hull will not produce the correct distance above hull values.

It should be noted that as the stripple and permustrain operations are not designed to manipulate the parent structure's composition, regardless of the specified genetic operation weights, an off-composition structure can be subject to crossover (for all search types) and permutomic and permucomp (for a VC search). For these structures, if the weight of the applicable genetic operations are zero, the crossover operation is chosen as the fallback option, while a warning message is printed in the run output to notify the user.

11 Construction of molecular units

For a MC or VC search, it is possible to use molecular units in building the unit cells just similar to the previous versions of the code (i.e., the case of FC search in the new version).

It should be noted that, however, in the new version of `XtalOpt` and regardless of the search type, the number of atoms of each type which can participate in construction of molecular units are limited to the smallest number of atoms of that type in the input list of chemical formulas.

For instance, given the following input:

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

then up to 2 and 3 atoms of Ti and O, respectively, can be used in the construction of molecular units.

12 Interface to machine-learning interatomic potentials

While there are a set of optimizers that are explicitly supported by `XtalOpt`, the output of an arbitrary optimizer (e.g., a machine learning interatomic potential) can be easily converted to that of a supported optimizer, using simple scripting. For instance, if the user sets the optimizer type to `VASP` while using an arbitrary code to perform local optimizations, the job file for the `XtalOpt` run would include the following steps: (i) converting VASP structure file (POSCAR) to the appropriate format for the user's code, (ii) perform the local optimization, and (iii) extract the results from the user's code and write `VASP` format output (i.e., OUTCAR and CONTCAR) files.

Such a workflow allows to benefit from the considerable speed-up that is offered by the machine learning potentials in a `XtalOpt` run. This is especially helpful in a VC search for multi-element systems where the entire composition space of the compound should be explored, involving possibly thousands of local optimizations, which is computationally prohibitive using first-principles approaches.

An easy-to-use Python script, `vasp_uip.py`, is included in the new release of the `XtalOpt` to facilitate such calculations. This wrapper, uses two of the recently developed universal interatomic potentials (UIPs), `MACE` [7] and `CHGNet` [8], to perform local structure optimization using the Atomic Simulation Environment (ASE) library [9]. The input/output format is that of `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 run parameters of this script can be adjusted through a series of command line options. A full list of available options can be obtained by:

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

and a simple energy, force, stress calculation can be performed with this script as:

```
python3 vasp_uip.py
```

given that a POSCAR file exists at the working directory and 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 efficient ground state search for a desired chemical system. It should be noted that the implemented potentials are 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., "objective" flag). There are also input flags that can parse multiple entries, while the entries might or might not involve multiple parameters (e.g., "elementalVolumes" and "seedStructures" flags).

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

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

```
seedStructures = structure1 , structure2
```

- In the case that a flag requires a single entry, and that entry has multiple parameters, the parameters should be a list of "space-separated" values, e.g., 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 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 of the above format.

13.2 Output files

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

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

Old filename	New filename
xtalopt-runtime-options.txt	cli-runtime-options.txt
xtaloptSettings.log	settings.log
xtaloptDebug.log	output.log

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

13.3 Verbose output

In the previous version of `XtalOpt`, if the code was compiled with the `cmake` flag

```
-DXTALOPT_DEBUG=ON
```

the output of the run would include a more extensive set of information, and this output would be saved to disk as the "xtaloptDebug.log" file (for both the CLI and GUI modes). In the new version, for simplicity and regardless of the compilation flag, the standard output of a CLI run will include all the additional information (e.g., details of hull and fitness calculations, calculated dot product of the RDF vectors, etc.) by setting the input flag:

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

This parameter has a default value of "false", can be modified during the run, and the output produced by this setting appears with a small indentation in the run output.

In the GUI mode, this option is available as a check box in the "Progress" tab (Figure 1c). However, and similar to the previous version, 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 is written.

This option is specifically designed to facilitate monitoring the run progress, e.g., by creating movies of the convex hull evolution during the run.

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

13.5 Legacy AFLOW-hardness optimization

The explicit support for AFLOW [10] hardness as an objective is removed. AFLOW hardness can still be used as an objective, just similar to any objective that should be maximized, using an interface executable script.

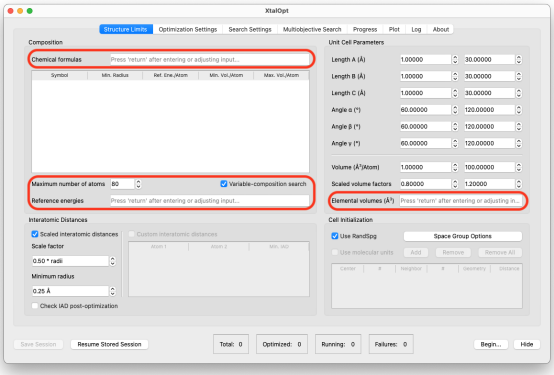
13.6 Backward compatibility

The new version of `XtalOpt` (version 14) does not read in the inupt 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.

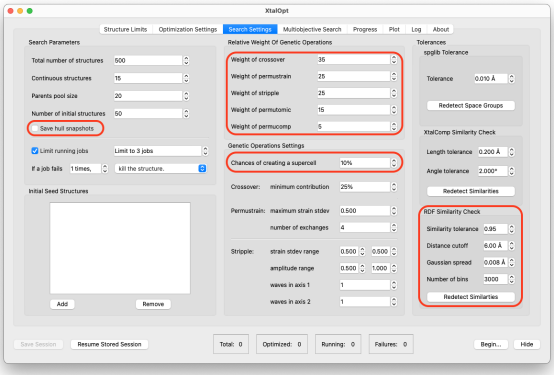
Appendix A New implementation in the GUI

The GUI of `XtalOpt` version 14 supports all the above mentioned developments. Generally, the format of the relevant input entries (e.g., chemical formulas, reference energies, elemental volumes) has the same format as those used in the CLI mode.

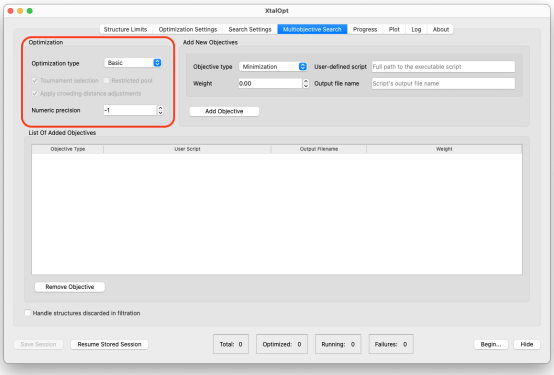
The GUI of the new version of code is redesigned to accommodate the new implementation. In particular, the "Structure Limits" and "Search Settings" tabs host the majority of the new settings, while the "Multi-objective Search" tab has a new section ("Optimization") where the user can choose the optimization type and adjust the relevant settings. Also, the GUI now includes a new "About" tab that offers some basic information and web links about the code. Figure 1 illustrate a few of tabs in the new GUI, with the changes relevant to the new implementation highlighted.



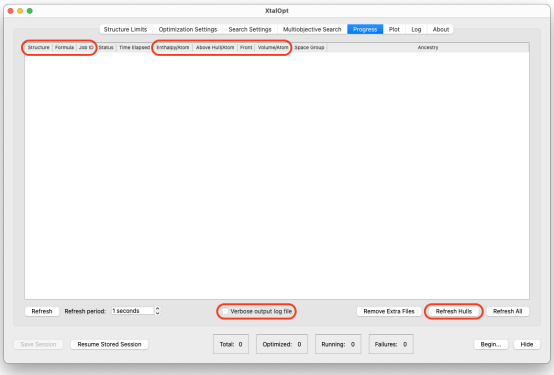
(a) The "Structure Limits" tab.



(b) The "Search Settings" tab.



(c) The "Multiobjective Search" tab.



(d) The "Progress" tab.

Figure 1: GUI tabs in `XtalOpt` version 14.

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

New flag	Comments (and *default values, if any)	Runtime adjustable
chemicalFormulas	Required input flag: composition of initial generation	No
vcSearch	Logical: perform variable-composition search (*false)	No
maxAtoms	Maximum unit cell size generated in the search (*20)	Yes
referenceEnergies	In local optimizer's units	No
weightPermutomic	Relative weight of applying permutomic (applies only if vcSearch=true)	Yes
weightPermucomp	Relative weight of applying permucomp (applies only if vcSearch=true)	Yes
randomSuperCell	A number 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	In $Å^3$ units	Yes
rdfTolerance	Threshold for similarity in [0.0, 1.0] (*0.0)	Yes
rdfCutoff	Maximum bond length considered in Å (*6.0)	Yes
rdfSigma	Spread of Gaussian in Å (*0.008)	Yes
rdfNumBins	Number of bins (*3000)	Yes
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	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	Yes
percentChanceStripple	weightStripple	Relative weight of applying operation	Yes
percentChancePermustrain	weightPermustrain	Relative weight of applying operation	Yes
objective		All entries are necessary	No
seedStructures		A comma-delimited list	No

Obsolete flag	Comments
empiricalFormula	Replaced with “chemicalFormulas” flag
formulaUnits	Merged into “chemicalFormulas” flag
usingFormulaUnitCrossovers	Now it is performed by default
usingOneGenePool	Now it is performed by default
usingSubcellMitosis	
usingMitoticGrowth	
chanceOfFutureMitosis	
mitosisDivisions	
mitosisA	
mitosisB	
mitosisC	
printSubcell	

References

[1] D. C. Lonie, E. Zurek, XtalOpt: An open-source evolutionary algorithm for crystal structure prediction, Comput. Phys. Commun. 182 (2) (2011) 372–387. doi:10.1016/j.cpc.2010.07.048.
URL https://linkinghub.elsevier.com/retrieve/pii/S0010465510003140

[2] S. Hajinazar, E. Zurek, XtalOpt version 13: Multi-objective evolutionary search for novel functional materials, Comput. Phys. Commun. 304 (2024) 109306. doi:10.1016/j.cpc.2024.109306.
URL https://www.sciencedirect.com/science/article/pii/S0010465524002297

[3] C. B. Barber, D. P. Dobkin, H. Huhdanpaa, The quickhull algorithm for convex hulls, ACM Trans. Math. Softw. 22 (4) (1996) 469–483. doi:10.1145/235815.235821.
URL https://doi.org/10.1145/235815.235821

[4] 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.
URL http://ieeexplore.ieee.org/document/996017/

[5] D. C. Lonie, E. Zurek, Identifying Duplicate Crystal Structures: XtalComp, An Open–Source solution, Comput. Phys. Commun. 183 (3) (2012) 690–697. doi:10.1016/j.cpc.2011.11.007.
URL http://www.sciencedirect.com/science/article/pii/S0010465511003699

[6] S. Hajinazar, A. Thorn, E. D. Sandoval, S. Kharabadze, A. N. Kolmogorov, MAISE: Construction of neural network interatomic models and evolutionary structure optimization, Comput. Phys. Commun. 259 (2021) 107679. doi:10.1016/j.cpc.2020.107679.
URL https://doi.org/10.1016/j.cpc.2020.107679

[7] I. Batatia, P. Benner, Y. Chiang, A. M. Elena, D. P. Kovács, J. Riebesell, et al., A foundation model for atomistic materials chemistry, arXiv (2023). doi:10.48550/arxiv.2401.00096.
URL http://arxiv.org/abs/2401.00096

[8] B. Deng, P. Zhong, K. J. Jun, J. Riebesell, K. Han, C. J. Bartel, 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.
URL https://www.nature.com/articles/s42256-023-00716-3

[9] A. Hjorth Larsen, J. JØrgen Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dułak, 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.
URL https://iopscience.iop.org/article/10.1088/1361-648X/aa680e

[10] S. Curtarolo, W. Setyawan, G. L. Hart, M. Jahnatek, R. V. Chepulskii, R. H. Taylor, et al., AFLOW: An automatic framework for high-throughput materials discovery, Comput. Mater. Sci. 58 (2012) 218–226. doi:10.1016/j.commatsci.2012.02.005.
URL https://linkinghub.elsevier.com/retrieve/pii/S0927025612000717