

XtalOpt 14.0 User Guide

(Document version 2)

Generated by Doxygen 1.9.8

1 XtalOpt Tutorial	1
1.1 Launch XtalOpt	1
1.2 Enter Composition and Restraints	1
1.2.1 Chemical Composition	1
1.2.2 Interatomic Distances (IAD)	2
1.2.3 Cell Parameters	2
1.2.4 Molecular-Unit Builder	3
1.2.5 Random Spacegroup Generator	3
1.3 Optimizer Setup	4
1.3.1 VASP	4
1.3.2 GULP	5
1.3.3 PWscf	6
1.3.4 CASTEP	7
1.3.5 SIESTA	8
1.3.6 MTP	10
1.3.7 Generic Optimizer	11
1.3.8 Machine Learning Potentials	13
1.4 Queue setup	14
1.4.1 Using a Remote PBS Cluster	15
1.4.2 Using a Remote SGE Cluster	16
1.4.3 Using a Remote SLURM Cluster	17
1.4.4 Using a Remote LSF Cluster	18
1.4.5 Using a Remote LoadLeveler Cluster	19
1.4.6 Running Optimizations Locally	20
1.5 What Is Written to the Local Directory?	20
1.5.1 Reading the results.txt File	21
1.6 Search Settings	22
1.7 Optimization Type and Multi-Objective Search	23
1.7.1 Specifying Objectives	23
1.7.2 Optimization Type	24
1.7.3 Multi-Objective Runs in the XtalOpt GUI	25
1.7.4 Filtering Structures: Constrained Search	26
1.7.5 Examples of User-defined Scripts	26
1.8 "Begin"	28
1.9 Monitor Progress	29
1.9.1 View Trends	30
1.9.2 View Crystals in Avogadro2	32
1.9.3 Plot a Simulated XRD Pattern	32
1.10 Command Line Interface	33

1.10.1 Multi-Objective Runs in the XtalOpt CLI	35
1.10.2 Submitting Optimization Jobs to a Queue in a Local Run	36
1.10.3 VASP "System" POTCAR	36
1.10.4 List of the input flags for the CLI setting file	36
2 Saving and Resuming Sessions in XtalOpt	41
2.1 How to Save Your Session	41
2.2 How to Resume Your Session	41
3 Optimization Schemes	42
3.1 Overview: What Are Optimization Schemes, and Why Use Them?	42
3.1.1 In a Nutshell...	42
3.1.2 More Details	43
3.2 Optimization Scheme User Interface	44
3.2.1 Optimization Step List	44
3.2.2 Add New Optimization Step	44
3.2.3 Remove Current Optimization Step	45
3.2.4 Select Template	45
3.2.5 Template Editor	45
3.2.6 Save Scheme	45
3.2.7 Resume Scheme	45
3.3 How to Build an Optimization Scheme?	45
3.4 How to Save an Optimization Scheme for Later?	46
3.5 How to Load an Optimization Scheme?	46
3.6 What is Saved?	46
3.7 Suggestions for Optimization Schemes	47
3.7.1 Crystals (XtalOpt)	47

1 XtalOpt Tutorial

1.1 Launch XtalOpt

Simply run the "XtalOpt" executable (or in MacOS, open the XtalOpt.app file).

1.2 Enter Composition and Restraints

The screenshot shows the XtalOpt application window with the 'Structure Limits' tab selected. The interface is divided into several sections:

- Composition:** Contains a text field for 'Chemical formulas' with the input 'Ti1O2 - Ti16O32, Ti4O4'. Below it is a table with columns: Symbol, Min. Radius, Ref. Ene./Atom, Min. Vol./Atom, and Max. Vol./Atom. The table contains two rows: O (Min. Radius: 0.33, Ref. Ene./Atom: -4.93563, Min. Vol./Atom: 1.20426, Max. Vol./Atom: 3.61278) and Ti (Min. Radius: 0.8, Ref. Ene./Atom: -7.78634, Min. Vol./Atom: 17.1573, Max. Vol./Atom: 51.4719). Below the table are fields for 'Maximum number of atoms' (set to 80) and 'Reference energies' (set to 'Ti3 -23.35901499, O16 -78.97000885'). There is a checkbox for 'Variable-composition search' which is checked.
- Unit Cell Parameters:** Contains fields for Length A (Å), Length B (Å), Length C (Å), Angle α (°), Angle β (°), Angle γ (°), Volume (Å³/Atom), Scaled volume factors, and Elemental volumes (Å³). Each field has a value and a range (e.g., Length A: 1.00000 to 30.00000).
- Interatomic Distances:** Contains a checkbox for 'Scaled interatomic distances' which is checked, and a checkbox for 'Custom interatomic distances' which is unchecked. Below these are fields for 'Scale factor' (set to 0.50 * radii) and 'Minimum radius' (set to 0.25 Å). There is also a checkbox for 'Check IAD post-optimization' which is unchecked.
- Cell Initialization:** Contains a checkbox for 'Use RandSpG' which is checked, and a checkbox for 'Use molecular units' which is unchecked. There are buttons for 'Add', 'Remove', and 'Remove All' under 'Space Group Options'. Below these are columns for 'Center', '#', 'Neighbor', '#', 'Geometry', and 'Distance'.

At the bottom of the window, there are buttons for 'Save Session', 'Resume Stored Session', and a status bar showing 'Total: 0', 'Optimized: 0', 'Running: 0', 'Failures: 0', and buttons for 'Begin...' and 'Hide'.

The interface opens to the "Structure Limits" tab, shown above.

1.2.1 Chemical Composition

The first field that one must set, is the "Chemical formulas" entry. In general, if the chemical formulas input involves entries of the same composition, XtalOpt will perform a fixed-composition search. As of XtalOpt 14, additional options are available: if they include different compositions, then the code performs a multi-composition search, i.e., restricting every new generated cell by the genetic operations to one of the compositions given in the input list entries. If, however, the "variable-composition" box is checked, regardless of the details of input chemical formulas, XtalOpt will perform a variable-composition search. This means that no restriction will be placed on the compositions generated by genetic operations, other than making sure that all chemical elements are present in the produced cell.

Here, we set the input list of chemical formulas as "Ti1O2 - Ti16O32, Ti4O4". This input instructs XtalOpt to create the first generation from the "Ti1O2" unit cell and its formula units from "1 - 16" and the "Ti4O4" cell composition. Further, by checking the variable-composition search box, we instruct the code to perform a variable-composition search.

Moreover, we set the maximum atom number to 80; so that no unit cell with a total number of atoms larger than this value will be produced by the code during the search. This is a measure introduced to manage the computational cost of the search.

Finally, since XtalOpt uses the distance above the convex hull to determine the energetic fitness, a set of reference energies can be introduced to be used as the reference points of the convex hull calculation. If no references are given, all elemental references with energies set to 0 will be used. If references are given, however, they must include all elements in the system. The input can be a comma-delimited list of entries, with each entry being the full chemical formula of a reference cell followed by its energy in the local optimizer's units. In this example we have previously calculated the values for the Ti3 and O16 cells as -23.35901499 and -78.97000885, respectively. Then, they are entered as the input string "Ti3 -23.35901499, O16 -78.97000885".

1.2.2 Interatomic Distances (IAD)

There are two different kinds of interatomic distances available: scaled interatomic distances and custom interatomic distances. If "Use Scaled Interatomic Distances" is checked, the covalent radii of the elements will be multiplied by the "Scale factor", and any radii below the "Minimum radius" will be set to the "Minimum radius." The minimum interatomic distance, then, between pairs of atoms in this setup is the sum of their radii. For our example, check the "Use Scaled Interatomic Distances" checkbox and set "Scale factor" to 0.40 and "Minimum radius" to 0.25.

Custom interatomic distances is an alternative option. If the "Use Custom Interatomic Distances" box is checked instead, the user can specify the minimum interatomic distance between every pair of atom types in the table below the checkbox.

Finally, a checkbox labelled "Check IAD Post-Optimization" is also available. If this box is checked, the interatomic distances are checked after the optimization is complete, and if any structures fail the interatomic distance check, they will be marked as failed structures and removed from the breeding pool.

1.2.3 Cell Parameters

We will assume that we know nothing about the system and use very loose restraints (however, note that a search is much more effective if chemically reasonable restraints are used). Set all cell length minima to 1 angstrom and maxima to 20 angstroms. Constrain the angles to be between 60 and 120 degrees, and the volume from 1 to 500 cubic angstroms. (Note that due to the angle adjustment described in CPC, 2011, 182, 372-387, 60-120 degrees is the largest range of cell angles that XtalOpt will generate.) Furthermore, the volume of the cell can be fixed, so that all cells generated will have the exact same volume.

As of version 13.0 of XtalOpt, a new option is added to aid in making a more educated guess for volume limits. This option can be utilized by setting the corresponding minimum and maximum scaling factors to appropriate "real numbers greater than zero" (e.g., 1.5 and 2.5 for the minimum and maximum values, respectively).

XtalOpt first calculates the total volume of spheres of covalent radii for all atoms in the unit cell. Then, it multiplies that total volume by the scaling factors to obtain the minimum and maximum limits of volume. The final calculated values are being updated in the corresponding volume minimum and maximum fields, as the user modifies the scaling factors.

XtalOpt 14 offers an option to set the volume limits for each element separately. In order to use this feature, one has to provide a comma-delimited list of entries, where each entry includes the full chemical formula of an elemental cell followed by its minimum and maximum volume limits (in $\text{\AA}^3/\text{cell}$ units). For example, for the Ti-O system "Ti3 50 100, O2 20 40" could be a valid input.

1.2.4 Molecular-Unit Builder

If the user chooses to define specific molecular units, this option will allow them to do so. Once the chemical compositions have been defined, the user can define a single-center molecule that assumes one of the VSEPR geometries. Since we might have a variety of compositions and unit cell sizes, in general, the maximum number of atoms of each type that can participate in building molecular units are the "smallest" atom count of that type in the input list of formulas. In our example, we can have molecular units with up to 1 Ti atom and 2 O atoms. We can also decide what the interatomic distance between the center and neighboring atoms will be. If there are left over atoms, they will be placed randomly after the molecular units are added.

The molecular units are only used in the initial generation, and cannot be used simultaneously with the [Random Spacegroup Generator](#).

1.2.5 Random Spacegroup Generator

Space Group	Possible Composition(s)	Allow randSpg?	Min xtals per Spg
180 P 62 2 2	O6Ti3	<input checked="" type="checkbox"/>	0
181 P 64 2 2	O6Ti3	<input checked="" type="checkbox"/>	0
182 P 63 2 2	O4Ti2,O8Ti4,O4Ti4	<input checked="" type="checkbox"/>	0
183 P 6 m m	O2Ti1,O4Ti2,O6Ti3,O8Ti4,O4Ti4	<input checked="" type="checkbox"/>	0
184 P 6 c c	O4Ti2,O8Ti4,O4Ti4	<input checked="" type="checkbox"/>	0
185 P 63 c m	O4Ti2,O8Ti4,O4Ti4	<input checked="" type="checkbox"/>	0
186 P 63 m c	O4Ti2,O8Ti4,O4Ti4	<input checked="" type="checkbox"/>	0
187 P -6 m 2	O2Ti1,O4Ti2,O6Ti3,O8Ti4,O4Ti4	<input checked="" type="checkbox"/>	0
188 P -6 c 2	O4Ti2,O8Ti4,O4Ti4	<input checked="" type="checkbox"/>	0
189 P -6 2 m	O2Ti1,O4Ti2,O6Ti3,O8Ti4,O4Ti4	<input checked="" type="checkbox"/>	0
190 P -6 2 c	O4Ti2,O8Ti4,O4Ti4	<input checked="" type="checkbox"/>	0
191 P 6/m m m	O2Ti1,O4Ti2,O6Ti3,O8Ti4,O4Ti4	<input checked="" type="checkbox"/>	0
192 P 6/m c c	O4Ti2,O8Ti4,O4Ti4	<input checked="" type="checkbox"/>	0
193 P 63/m c m	O4Ti2,O8Ti4,O4Ti4	<input checked="" type="checkbox"/>	0
194 P 63/m m c	O4Ti2,O8Ti4,O4Ti4	<input checked="" type="checkbox"/>	0
195 P 2 3	O4Ti2,O6Ti3,O8Ti4,O4Ti4	<input checked="" type="checkbox"/>	0
196 F 2 3	O8Ti4,O4Ti4	<input checked="" type="checkbox"/>	0
197 I 2 3		<input type="checkbox"/>	0
198 P 21 3	O8Ti4,O4Ti4	<input checked="" type="checkbox"/>	0
199 I 21 3		<input type="checkbox"/>	0
200 P m 3	O6Ti3,O8Ti4,O4Ti4	<input checked="" type="checkbox"/>	0

Select all Deselect all Increment All Decrement All

With the implementation of RandSpg, the initial generation of structures can be created by using specific spacegroups (or a variety of spacegroups). The choice for spacegroups will be limited by the details of the input chemical formulas.

Spacegroup constraints can only be used in the initial generation. Randspg cannot be used simultaneously with [Molecular-Unit Builder](#).

For more information on RandSpg, see [this paper](#).

1.3 Optimizer Setup

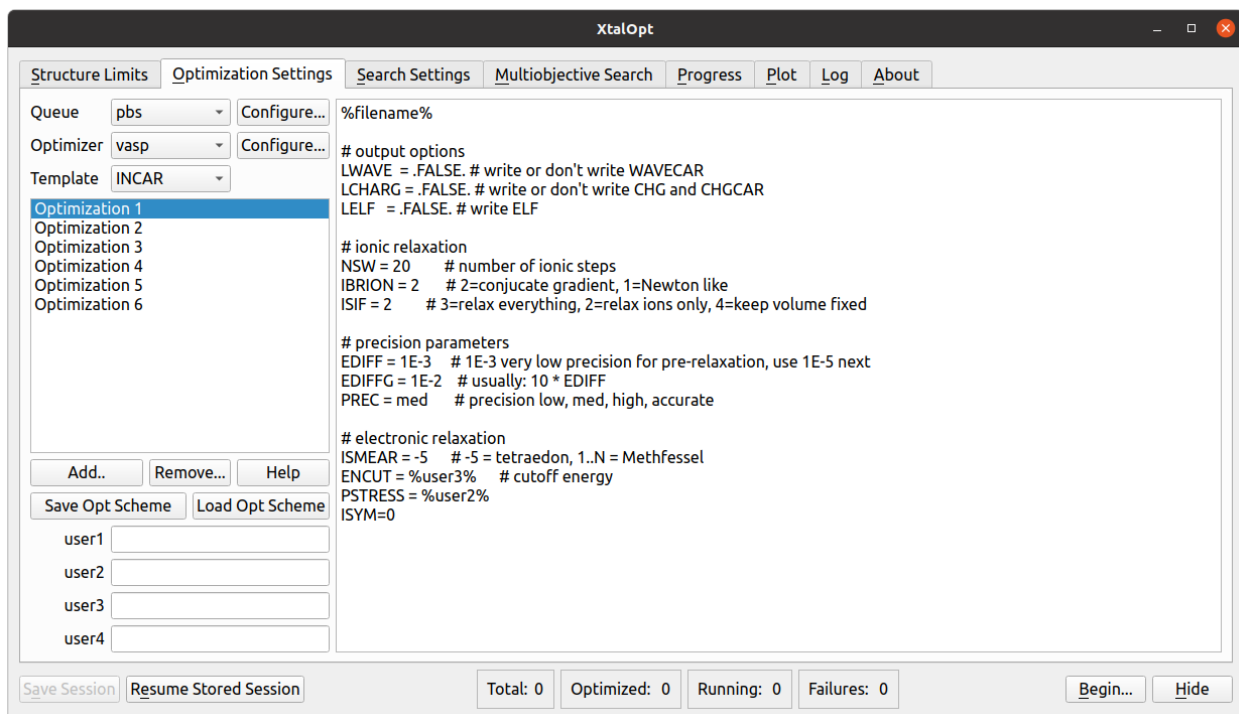
XtalOpt primarily supports the [VASP](#), [GULP](#), [PWscf](#), [CASTEP](#), [SIESTA](#), and [MTP](#) platforms for performing geometry optimizations, as well as desired [Machine Learning Potentials](#) through scripting. Each is detailed in its own section below.

As of release 12 XtalOpt has supported a generic optimizer, which can potentially be used for many different kinds of optimizers. The generic optimizer is unique in many ways, and details on how to use it are given in the [Generic Optimizer](#) section.

New to release 12 as well, a different optimizer may be used for each optimization step. Simply click on the optimization step in the "Optimization Settings" tab before selecting the optimizer for that step.

Be aware that the submit files included with the schemes discussed below are configured to work on the Zurek group's cluster at SUNY Buffalo's Center for Computational Research. It may take some experimentation to get jobs to submit successfully, and you may need to contact the system administrators of the cluster for assistance for information about MPI, executable locations, etc. Perhaps the easiest method to construct a submit script that works for your specific situation is to run some trial submissions by hand, and then replace the structure/search specific information with the appropriate keywords once a working script has been generated.

1.3.1 VASP



On the next tab, load the optimization scheme by clicking the "Load Opt Scheme" button and selecting the "schemes/vasp-xtalopt.scheme" file that is distributed with the source code. If you do not have a copy of the source code, the scheme file can be obtained by clicking [here](#).

For more details on optimization schemes, see [Optimization Schemes](#).

Take a moment to look through each file for each optimization step. Notice that the INCAR template includes two user-specified values, %user2% and %user3% for the external pressure and the energy cutoff, respectively. By entering appropriate values in the "user2:" and "user3:" fields on the left, it is easy to update these values for all optimization steps.

Notice the other %keyword% values in the job.pbs templates. These are used to enter information that is specific to a search or structure when the actual input files are written prior to job submission. Click the "Help" button for a full listing of the available keywords.

For the POTCAR templates, the keyword %fileContents:/path/to/file% is to be used for each POTCAR template. This is performed like so:

```
%fileContents:/path/to/first/potcar/in/alphabetical/order%
%fileContents:/path/to/second/potcar/in/alphabetical/order%
```

The regions enclosed by the "%%" signs will be replaced with the contents of the files. The POTCAR files need to be in alphabetical order (based on the element symbol) because XtalOpt will order the elements in the POSCAR file in alphabetical order as well.

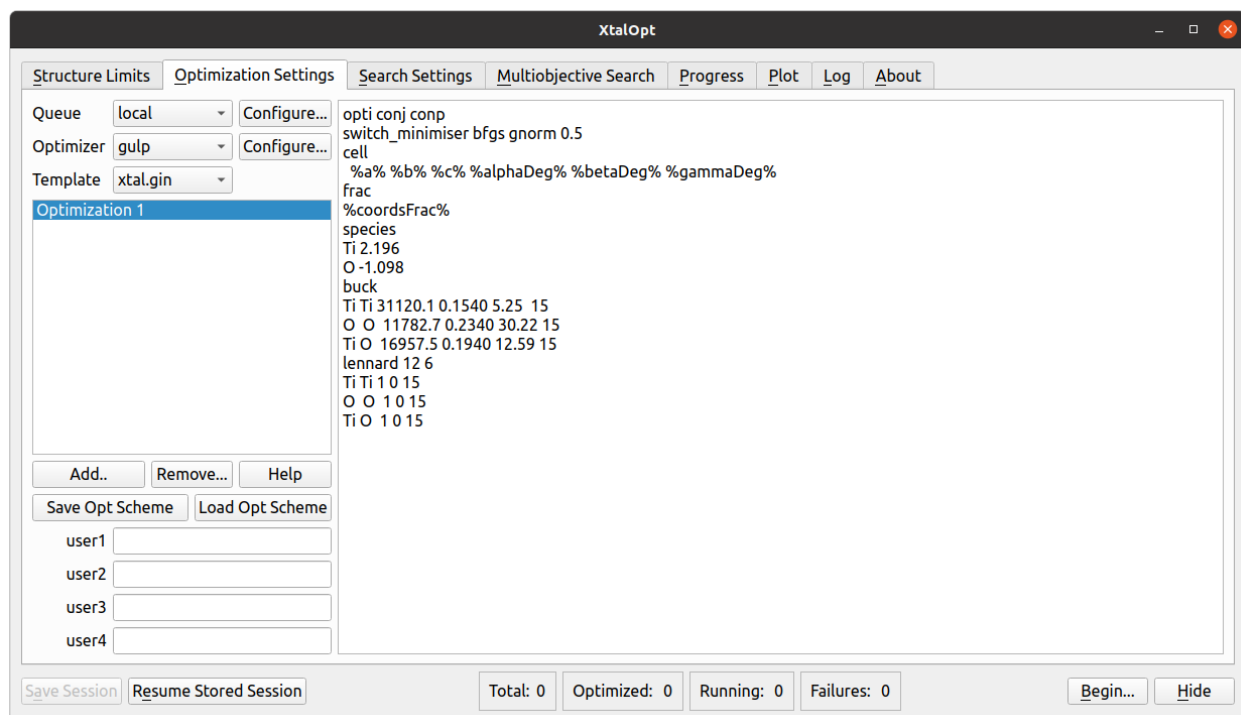
It is necessary to have the VASP POTCAR files for each atomic species located somewhere on the local computer. See the VASP manual for information on obtaining the POTCAR files.

XtalOpt expects VASP to use the default filenames, mainly POSCAR, CONTCAR, and OUTCAR.

Moreover, XtalOpt version 13.0 supports the OUTCAR files produced by VASP machine learning force fields.

[Skip to next section.](#)

1.3.2 GULP



On the next tab we choose GULP for the local optimizer and enter a template for GULP to use. Select "GULP" as the "Optimizer" and "xtal.gin" as "Template". Next, fill out the text field on the right with the following template:

```
opti conj conp
switch_minimiser bfgs gnorm 0.5
cell
  %a% %b% %c% %alphaDeg% %betaDeg% %gammaDeg%
frac
%coordsFrac%
species
Ti 2.196
O -1.098
buck
Ti Ti 31120.1 0.1540 5.25 15
O O 11782.7 0.2340 30.22 15
Ti O 16957.5 0.1940 12.59 15
lennard 12 6
Ti Ti 1 0 15
O O 1 0 15
Ti O 1 0 15
```

Alternatively, one can load the scheme file distributed with the source code under schemes/gulp-TiO-xtalopt.scheme. If the source code is not available, the scheme file can be obtained by clicking [here](#).

For more details on optimization schemes, see [Optimization Schemes](#).

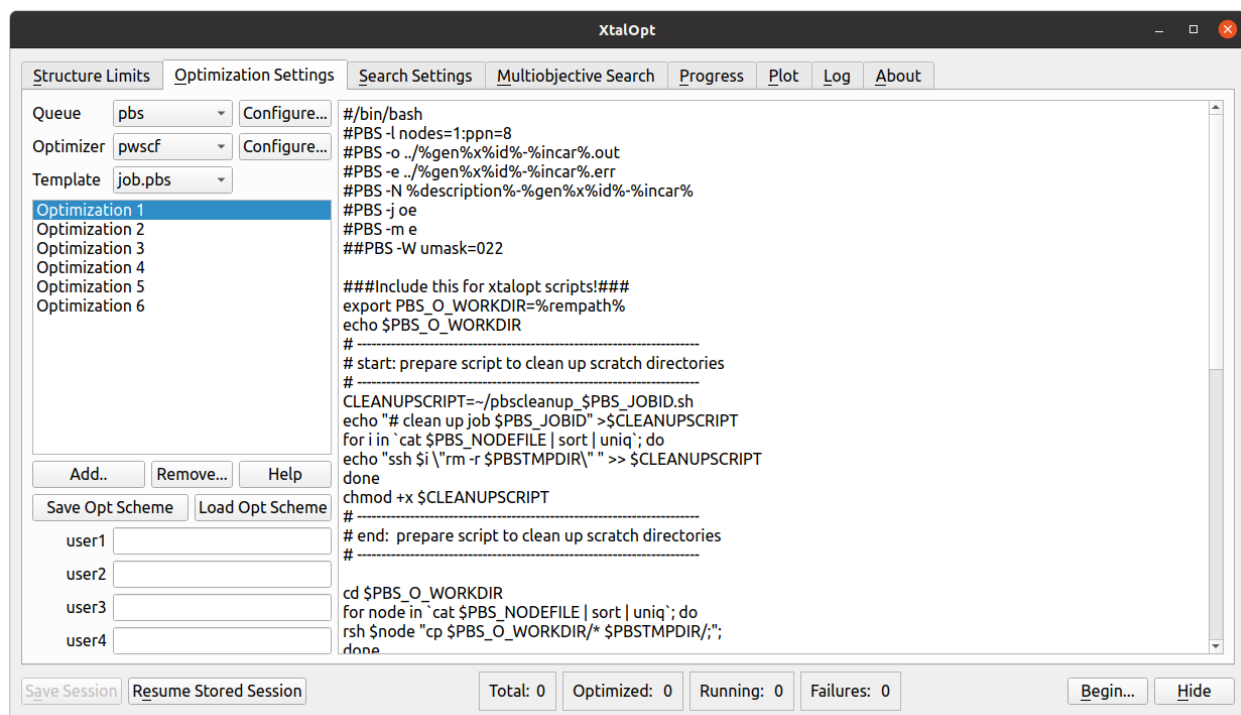
Note the "%" surrounding various keywords. These will be replaced by the structure-specific data when the optimizer is invoked for each structure. Click "Help" to view all of the keywords available. The number of optimization steps can be modified with the "Add/Resume" buttons. The "user" fields in the lower left corner allow users to specify their own keyword/value pairs, which is useful for making changes to multiple optimization steps at once. We will only be using one optimization step in this tutorial.

XtalOpt expects GULP to use the following filenames:

```
gulp < xtal.gin > xtal.got
```

[Skip to next section.](#)

1.3.3 PWscf



On the next tab, load the optimization scheme that is distributed with the source code under the `schemes/` directory. The scheme that we want is named "pwsf-xtalopt.scheme". If the source code is not available, the scheme file can be obtained by clicking [here](#).

For more details on optimization schemes, see [Optimization Schemes](#).

Each PWscf input file will need to be edited to specify:

1. The `pseudo_dir` containing the pseudopotential files on the remote cluster, and
2. The pseudopotentials for each atom (under `ATOMIC_SPECIES`)

Take a moment to look through each file for each optimization step.

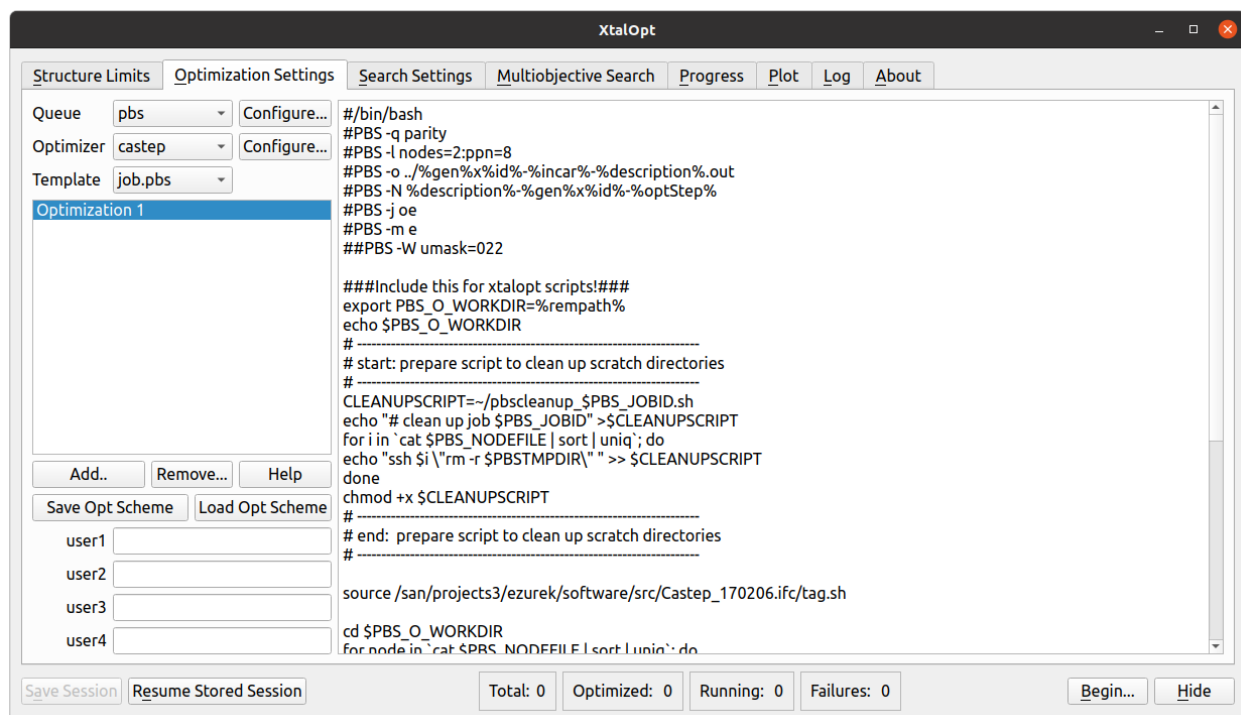
Notice the `%keyword%` values in the `job.pbs` templates. These are used to enter information that is specific to a search or structure when the actual input files are written prior to job submission. Click the "Help" button for a full listing of the available keywords.

XtalOpt expects PWscf to use the following filenames:

```
pw.x < xtal.in > xtal.out
```

[Skip to next section.](#)

1.3.4 CASTEP



On the next tab, load the optimization scheme that is distributed with the source code under the `schemes/` directory. The scheme that we want is named "castep-xtalopt.scheme". If the source code is not available, the scheme file can be obtained by clicking [here](#).

For more details on optimization schemes, see [Optimization Schemes](#).

It is important to note that CASTEP input files require the "%" character to define blocks. The percent character is special in the XtalOpt input template parser to define keywords (see below). To insert a literal "%" into the input, use %percent%.

E. g., specification of the fractional coordinate block in the .cell template should look like:

```
%block% POSITIONS_FRAC
%coordsFrac%
%endblock% POSITIONS_FRAC
```

Take a moment to look through each file for each optimization step.

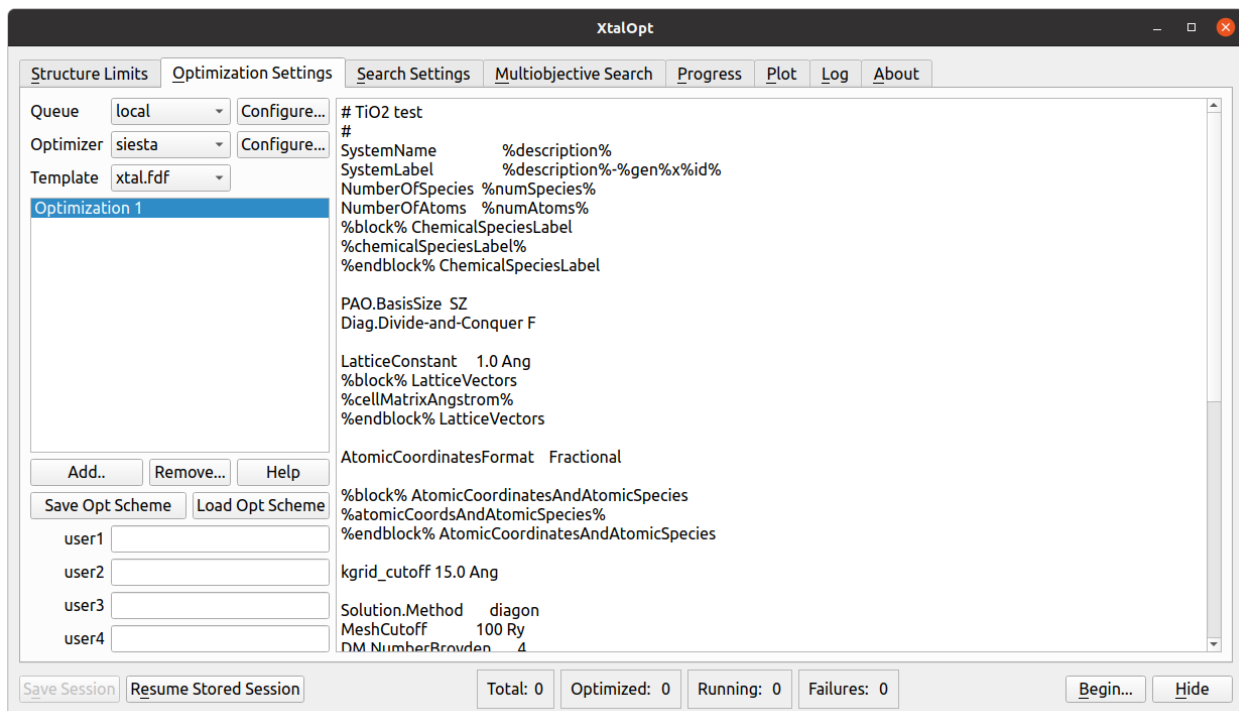
Notice the %keyword% values in the job.pbs templates. These are used to enter information that is specific to a search or structure when the actual input files are written prior to job submission. Click the "Help" button for a full listing of the available keywords.

XtalOpt expects CASTEP to use the following filenames:

```
# XtalOpt will write xtal.cell, xtal.param
castep xtal
# CASTEP will create xtal.castep
```

[Skip to next section.](#)

1.3.5 SIESTA



On the next tab we choose SIESTA for the local optimizer and enter a template for SIESTA to use. Select "SIESTA" as the "Optimizer" and "xtal.fdf" as "Template".

Next, fill out the text field on the right with the following template:

```
# TiO2 test
#
SystemName      %description%
SystemLabel     %description%-%gen%x%id%
NumberOfSpecies %numSpecies%
NumberOfAtoms   %numAtoms%
%block% ChemicalSpeciesLabel
%chemicalSpeciesLabel%
%endblock% ChemicalSpeciesLabel

PAO.BasisSize  SZ
Diag.Divide-and-Conquer F

LatticeConstant 1.0 Ang
%block% LatticeVectors
%cellMatrixAngstrom%
%endblock% LatticeVectors

AtomicCoordinatesFormat Fractional

%block% AtomicCoordinatesAndAtomicSpecies
%atomicCoordsAndAtomicSpecies%
%endblock% AtomicCoordinatesAndAtomicSpecies

kgrid_cutoff 15.0 Ang

Solution.Method      diagon
MeshCutoff           100 Ry
DM.NumberBroyden      4
DM.UseSaveDM          T
DM.MixingWeight       0.1      # New DM amount for next SCF cycle
DM.Tolerance          1.d-3    # Tolerance in maximum difference
                        # between input and output DM
MaxSCFIterations      20

WriteCoorStep        .true.
WriteForces           .true.

XC.functional         GGA
XC.authors            PBE

MD.TypeOfRun          Broyden
MD.Variable-Cell       T
MD.Target-pressure     0.0 GPa
MD.Num-CG-steps        30
MD.Max-Stress-Tol      2.0 GPa

MD.Broyden.Initial.Inverse.Jacobian 1.0
MD.Broyden.History.Steps 6

%copyFile:/path/to/first/psf%
%copyFile:/path/to/second/psf%
```

Or load the optimization scheme by clicking the "Load Opt Scheme" button and selecting the "schemes/siesta-TiO-xtalopt.scheme" file that is distributed with the source code. If the source code is not available, the scheme file can be obtained by clicking [here](#).

For more details on optimization schemes, see [Optimization Schemes](#).

For SIESTA, it is required that a ".psf" file (a pseudopotential file) be present for each element. This can be done using the %copyFile:/path/to/file% keyword in the "xtal.fdf" template like so:

```
%copyFile:/path/to/first/psf/file%
%copyFile:/path/to/second/psf/file%
```

The specified files will be copied to the structure's directory, and the region between the "%%" signs will be removed from the final "xtal.fdf" file. See the SIESTA manual for information on obtaining the psf files.

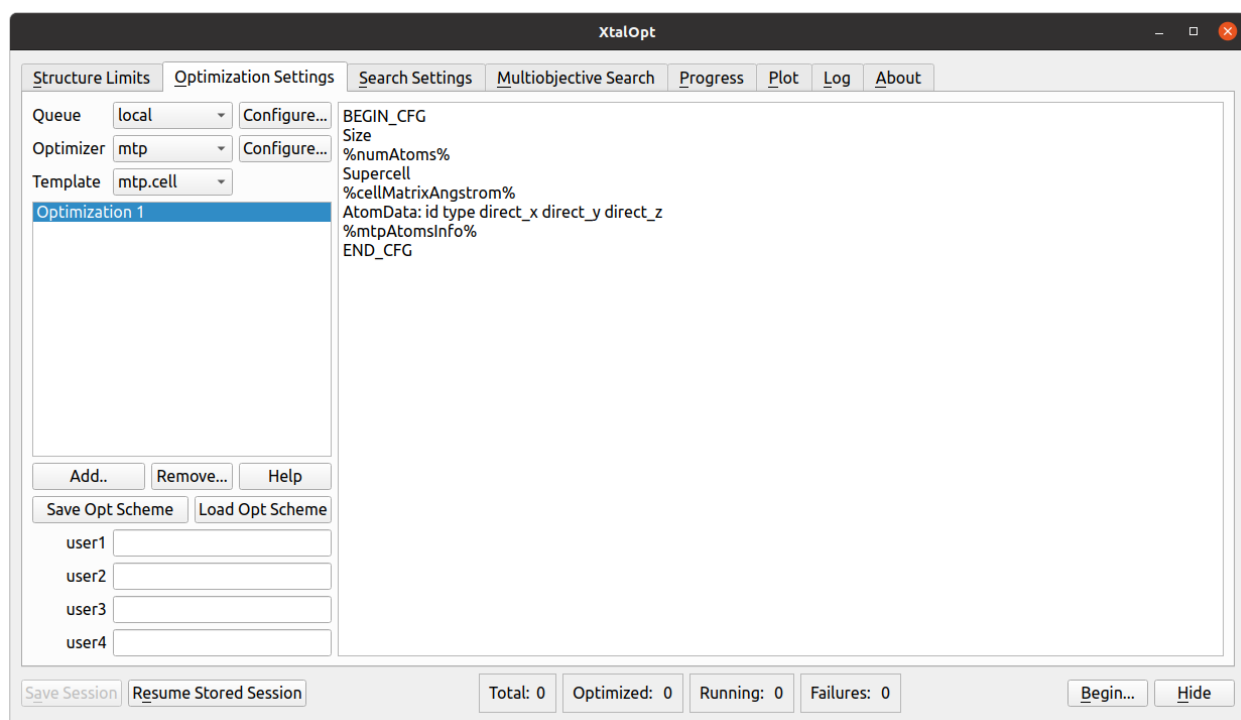
Notice the other %keyword% values in the xtal.fdf templates. These are used to enter information that is specific to a search or structure when the actual input files are written prior to job submission. Click the "Help" button for a full listing of the available keywords.

Note that in the current implementation XtalOpt uses the "Total Final Energy" printed in the output to calculate the convex hull and determine the fitness of a structure. If the user would like to use a different thermodynamic quantity for the fitness, please contact the XtalOpt developers.

XtalOpt expects SIESTA to use the following filenames:

```
siesta < xtal.fdf > xtal.out
```

1.3.6 MTP



As of XtalOpt 14, the code supports **moment tensor potentials** (MTP) for local optimizations, which for short, we refer to this as the MTP optimizer in this document.

Using the MTP optimizer will require an existing binary of the "MLIP code" to be set as the optimizer, and three template entries, i.e., a potential file, a setting file template, and a unit cell template that will be written as the main structure input for MTP optimizer at the runtime.

In general, the user must have a pre-trained potential for the desired system, and can introduce it in the potential field using:

```
%fileContents:/Users/sam/XtalOpt/mtp_pot%
```

or by directly entering the potential file's text in the corresponding box. For relaxation settings, the entry is simply:

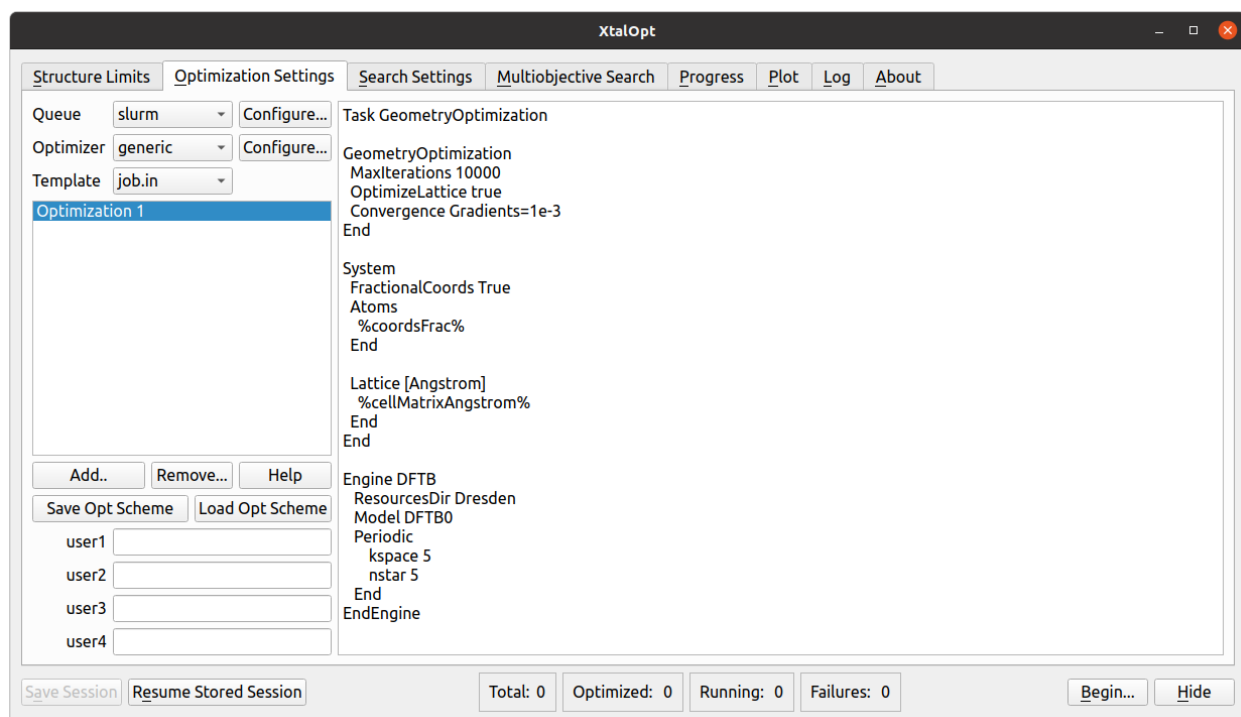
mtp-filename mtp.pot

and for the cell template, using the XtalOpt keywords, the following input will result in a proper input file for the MTP optimizer:

```
BEGIN_CFG
Size
%numAtoms%
Supercell
%cellMatrixAngstrom%
AtomData: id type direct_x direct_y direct_z
%mtpAtomsInfo%
END_CFG
```

As an example, the scheme file "mtp-LiSi-xtalopt.scheme" can be obtained from the source code, or if the source code is not available by clicking [here](#), which has ready to use templates for unit cell and relaxation settings.

1.3.7 Generic Optimizer



The "Generic Optimizer" can use many different kinds of optimizers. A certain set of rules must be followed, though, for an optimizer to be compatible. Below are the rules of the generic optimizer.

1. XtalOpt will only generate one input file with keywords, and it will be named "job.in". This file can be renamed with the job script. Other input files can also be created or copied in the job script (there is also a keyword called %copyfile:% that tells XtalOpt to copy a file into the destination directory). But only one "job.in" file will be generated by XtalOpt.
2. The main output file needs to ultimately be named "job.out". As with the "job.in" file, the normal output files can be renamed in the job script, and other output files can exist in the directory. But XtalOpt will only try to read a "job.out" file.

3. The "job.out" file must be a registered ".out" format in Open Babel. Open Babel is used to automatically detect and read the format of the "job.out" file, so Open Babel must be able to read it correctly. Since Open Babel is an open-source project, new ".out" formats can be added if Open Babel does not already have it.

As long as the above 3 rules are followed, just about any optimizer can be used for the generic optimizer. In our example here, we will be using DFTB from ADF 2018. Instructions are as follows:

On the next tab we choose generic for the local optimizer and enter a template for the optimizer to use. Select "generic" as the "Optimizer" and "job.in" as "Template".

We are going to use DFTB from ADF 2018 as an example. There is also a sample scheme for DFTB from ADF 2017 (located here "schemes/generic-adf-dftb-2017-xtalopt.scheme" in the XtalOpt source directory) if ADF 2018 is unavailable.

Next, fill out the text field on the right with the following template:

```
Task GeometryOptimization

GeometryOptimization
  MaxIterations 10000
  OptimizeLattice true
  Convergence Gradients=1e-3
End

System
  FractionalCoords True
  Atoms
    %coordsFrac%
  End

  Lattice [Angstrom]
    %cellMatrixAngstrom%
  End
End

Engine DFTB
  ResourcesDir Dresden
  Model DFTB0
  Periodic
    kspace 5
    nstar 5
  End
EndEngine
```

Or load the optimization scheme by clicking the "Load Opt Scheme" button and selecting the "schemes/generic-adf-dftb-2018-xtalopt.scheme" file that is distributed with the source code. If the source code is not available, the scheme file can be obtained by clicking [here](#).

For more details on optimization schemes, see [Optimization Schemes](#).

There are many %keyword% values available for the generic optimizer. These are used to enter information that is specific to a search or structure when the actual input files are written prior to job submission. Click the "Help" button for a full listing of the available keywords.

XtalOpt expects the generic optimizer to use the following filenames:

```
<optimizer> < job.in > job.out
```

In our case with ADF 2018, the optimizer will be the "ams" executable located in the ADF bin directory. Make sure the ADF environment variables are set up, and call it like so:

```
$ADFBIN/ams < job.in > job.out
```

As described in the rules for the generic optimizer above, a "job.in" file is created by XtalOpt, and this is to be used for the input. XtalOpt will try to read "job.out" when the job is complete, so the output must be named "job.out." Open Babel is responsible for determining the type of file of "job.out" and reading the values correctly.

Important Note: For the generic optimizer, XtalOpt will consider the optimization to be a success if Open Babel reads atoms, a unit cell, and an energy/enthalpy from the file. However, Open Babel being able to read these components does not always mean that the optimization was a success. To ensure that XtalOpt does not report a failed optimization as a success, a user can add something like this to the end of their job script:

```
COMPLETION_STRING="Some string that indicates the job succeeded"
if grep -Fq "$COMPLETION_STRING" job.out
then
    # The completion string was found
    echo "Completion string was found!"
else
    # The completion string was not found
    # XtalOpt only checks for job.out, so rename it to cause an error
    echo "Completion string was not found!"
    mv job.out job_failed.out

    # The presence of a job.out file may indicate to XtalOpt that the
    # job is complete if XtalOpt is unable to check a job's status.
    touch job.out
fi
```

The "\$COMPLETION_STRING" here is some string in the output file that is only present if the optimization was successful.

Renaming the "job.out" file at the end of the job script will cause the job to fail since XtalOpt will only try to read "job.out". This can ensure that only successful optimizations will be reported as "Optimized", and the rest will fail.

1.3.8 Machine Learning 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: (1) converting VASP structure file (POSCAR) to the appropriate format for the user's code, (2) perform the local optimization, and (3) 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 modern machine learning potentials in an XtalOpt run. This is especially helpful in variable-composition search for multi-element systems where the entire composition space of the compound should be explored, involving possibly thousands of local optimizations, which is prohibitive using first-principles approaches.

An example, "vasp_uip.py", a user-friendly Python script to perform such a task using the universal interatomic potentials **MACE** and **CHGNet** is available with the XtalOpt 14 release. This script simply mimics the VASP behavior in reading a POSCAR file and producing CONTCAR and OUTCAR files after local optimization, and hence it can be used when VASP is chosen as the optimizer in XtalOpt. It also offers a range of options to customise the local optimization, which can be easily obtained by:

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

It should be noted that the choice of MACE and CHGNet was mostly to demonstrate the possibility of the workflow. Any such a platform can be easily added to the this script by a user who is familiar with Python scripting.

1.4 Queue setup

XtalOpt currently supports using the [PBS](#), [SGE](#), [SLURM](#), [LSF](#), and [LoadLeveler](#) queuing systems on remote SSH-accessible clusters, as well as an internal [local](#) queue that manages calculations on the user's workstation. Each queueing interface is detailed in its own section below.

As of release 12, a different queue interface (and a different optimizer) can be used for each optimization step. This can be particularly useful if a quick optimization is to be performed on the local computer before further optimizations are performed on the cluster.

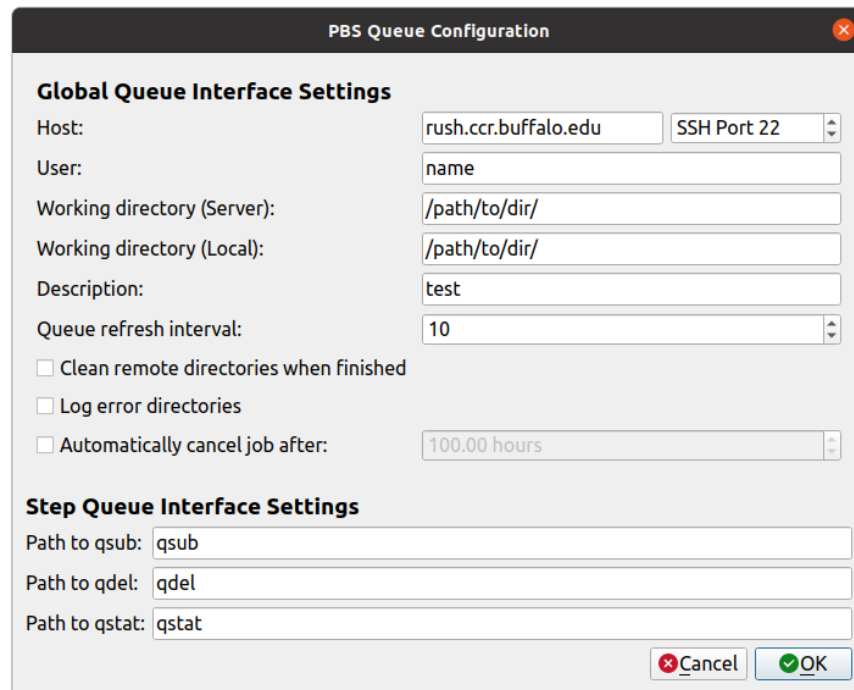
In addition, the queue configure menu (accessed via the "Configure..." button next to the queue interface) is now divided into two different parts: global queue interface settings and step queue interface settings. The global queue interface settings are used for all optimization steps. The step queue interface settings are for a particular optimization step. The global queue interface settings are as follows:

- host: The hostname of the cluster's head node.
- user: The username used to log into the cluster.
- Working directory (Server): A directory that is readable/writable by "user" on the cluster, used when performing optimizations.
- Working directory (Local): A directory that is readable/writable by the current user on the local computer. This is where the final structures and resume files are written.
- Description: Used for the %description% keyword in input templates.
- Queue refresh interval: The number of seconds to wait before querying the jobs' statuses again with the queue interface.
- Clean remote directories when finished: will remove all of the generated files from the cluster. Only the files on the local computer will be kept. If you do not want this to occur, make sure to uncheck this option.
- Log error directories: if checked, structures that produce an error will have their directory saved within an "error← Dirs" directory in the local working directory. The structure's directory will be named "<generation>x<id← Number>". This setting can be useful for debugging errors. Note that a structure will overwrite its error directory if another error occurs.
- Automatically cancel job after: if checked, a job will be killed if the specified number of hours are exceeded. The time checked is queue time + running time. This can be useful for cases where XtalOpt can't check whether a job is running or not, and too great a time has been exceeded. It can also be useful for optimizers with bugs that occasionally cause them to run forever.

To change the step queue interface settings at a particular step, first select a step in the "Optimization Settings" tab, and then click "Configure..." next to the queue interface.

A description for each of the different queue interfaces now follows.

1.4.1 Using a Remote PBS Cluster



The image shows a 'PBS Queue Configuration' dialog box. It is divided into two sections: 'Global Queue Interface Settings' and 'Step Queue Interface Settings'. The 'Global' section includes fields for Host (rush.ccr.buffalo.edu), User (name), Working directory (Server) (/path/to/dir/), Working directory (Local) (/path/to/dir/), Description (test), Queue refresh interval (10), and three checkboxes: 'Clean remote directories when finished', 'Log error directories', and 'Automatically cancel job after: 100.00 hours'. The 'Step' section includes fields for Path to qsub (qsub), Path to qdel (qdel), and Path to qstat (qstat). At the bottom right are 'Cancel' and 'OK' buttons.

Select "PBS" from the list of Queues, and then click the "Configure..." button. The step queue interface settings are:

- Path to qsub: Where to find the qsub executable on the remote cluster. Note that if qsub is in the cluster's \$PATH, setting this to just 'qsub' will work.
- Path to qdel: Where to find the qdel executable on the remote cluster. Note that if qdel is in the cluster's \$PATH, setting this to just 'qdel' will work.
- Path to qstat: Where to find the qstat executable on the remote cluster. Note that if qstat is in the cluster's \$PATH, setting this to just 'qstat' will work.

A new template, "job.pbs" is added to the list of available templates. This is the job submission script for PBS. This script should roughly follow this design:

```
#!/bin/bash
#PBS -l nodes=1:ppn=8
#PBS -o ../%gen%xid%-%optstep%.out
#PBS -e ../%gen%xid%-%optstep%.err
#PBS -N %description%-%gen%xid%-%optstep%

###Include this for XtalOpt scripts!###
export PBS_O_WORKDIR=%rempath%

# Change to structure's working directory, copy input files to node's scratch dirs:
for node in `cat $PBS_NODEFILE | sort | uniq`; do
  rsh $node "cp $PBS_O_WORKDIR/* $PBSTMPDIR/";
done

# Move to the scratch directory
cd $PBSTMPDIR
echo "running in directory $PBSTMPDIR"

# Set any environment variables needed for the optimizer/MPI here:

# Run optimizer, be sure to use the filenames that XtalOpt expects.
```

```
# See the template menu in XtalOpt and the example templates in the
# schemes/ directory of the XtalOpt sources.

# Don't forget to clean up after MPI if needed!

// Print files from each node
for node in `cat $PBS_NODEFILE | sort | uniq`; do
echo "$node:"
rsh $node "ls -l $PBSTMPDIR"
done
# Copy back results from master node's scratch directory
cp $PBSTMPDIR/* $PBS_O_WORKDIR/
```

A handy trick for monitoring jobs outside of XtalOpt is to include the following line in job.pbs:

```
#PBS -N %description%-%gen%x%id%-%optstep%
```

This will name each job, for example, xtalSearch-3x4-2, where xtalSearch is a user-specified description of the search, and 3x4-2 means that it is the fourth structure in the third generation running its second optimization step.

It may take some experimentation to get jobs to submit successfully, and you may need to contact the system administrators of the cluster for assistance or information about MPI, executable locations, etc. Perhaps the easiest method to find the correct PBS script is to run some trial submissions by hand, and then replace the structure/search specific information with the appropriate keywords once a working script has been generated.

For more details on optimization schemes, see [Optimization Schemes](#).

[Skip to next section.](#)

1.4.2 Using a Remote SGE Cluster

SGE Queue Configuration

Global Queue Interface Settings

Host: rush.ccr.buffalo.edu SSH Port 22

User: name

Working directory (Server): /path/to/dir/

Working directory (Local): /path/to/dir/

Description: test

Queue refresh interval: 10

☐ Clean remote directories when finished

☐ Log error directories

☐ Automatically cancel job after: 100.00 hours

Step Queue Interface Settings

Path to qsub: qsub

Path to qdel: qdel

Path to qstat: qstat

Cancel OK

Select "SGE" from the list of Queues, and then click the "Configure..." button. The step queue interface settings are:

- Path to qsub: Where to find the qsub executable on the remote cluster. Note that if qsub is in the cluster's \$PATH, setting this to just 'qsub' will work.
- Path to qdel: Where to find the qdel executable on the remote cluster. Note that if qdel is in the cluster's \$PATH, setting this to just 'qdel' will work.
- Path to qstat: Where to find the qstat executable on the remote cluster. Note that if qstat is in the cluster's \$PATH, setting this to just 'qstat' will work.

A new template, "job.sge" is added to the list of available templates. This is the job submission script for SGE. It may take some experimentation to get jobs to submit successfully, and you may need to contact the system administrators of the cluster for assistance or information about MPI, executable locations, etc. Perhaps the easiest method to find the correct SGE script is to run some trial submissions by hand, and then replace the structure/search specific information with the appropriate keywords once a working script has been generated.

For more details on optimization schemes, see [Optimization Schemes](#).

[Skip to next section.](#)

1.4.3 Using a Remote SLURM Cluster

SLURM Queue Configuration

Global Queue Interface Settings

Host: SSH Port

User:

Working directory (Server):

Working directory (Local):

Description:

Queue refresh interval:

☐ Clean remote directories when finished

☐ Log error directories

☐ Automatically cancel job after:

Step Queue Interface Settings

Path to sbatch:

Path to scancel:

Path to squeue:

Select "SLURM" from the list of Queues, and then click the "Configure..." button. The step queue interface settings are:

- Path to sbatch: Where to find the sbatch executable on the remote cluster. Note that if sbatch is in the cluster's \$PATH, setting this to just 'sbatch' will work.
- Path to scancel: Where to find the scancel executable on the remote cluster. Note that if scancel is in the cluster's \$PATH, setting this to just 'scancel' will work.

- Path to squeue: Where to find the squeue executable on the remote cluster. Note that if squeue is in the cluster's \$PATH, setting this to just 'squeue' will work.

A new template, "job.slurm" is added to the list of available templates. This is the job submission script for SLURM. It may take some experimentation to get jobs to submit successfully, and you may need to contact the system administrators of the cluster for assistance or information about MPI, executable locations, etc. Perhaps the easiest method to find the correct SLURM script is to run some trial submissions by hand, and then replace the structure/search specific information with the appropriate keywords once a working script has been generated.

For more details on optimization schemes, see [Optimization Schemes](#).

[Skip to next section.](#)

1.4.4 Using a Remote LSF Cluster

LSF Queue Configuration

Global Queue Interface Settings

Host: rush.ccr.buffalo.edu SSH Port 22

User: name

Working directory (Server): /path/to/dir/

Working directory (Local): /path/to/dir/

Description: test

Queue refresh interval: 10

☐ Clean remote directories when finished

☐ Log error directories

☐ Automatically cancel job after: 100.00 hours

Step Queue Interface Settings

Path to bsub: bsub

Path to bkill: bkill

Path to bjobs: bjobs

Cancel OK

Select "LSF" from the list of Queues, and then click the "Configure..." button. The step queue interface settings are:

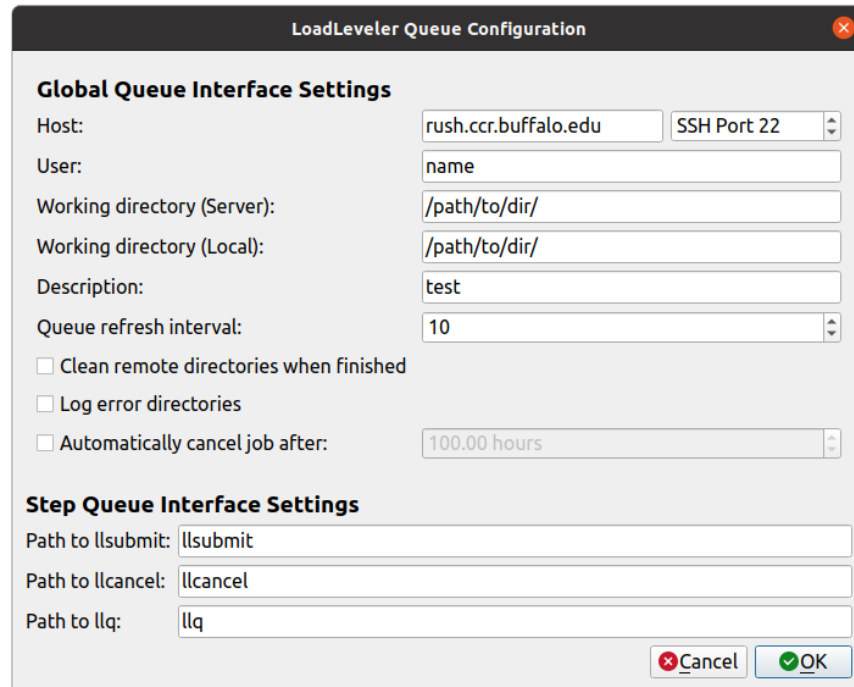
- Path to bsub: Where to find the bsub executable on the remote cluster. Note that if bsub is in the cluster's \$PATH, setting this to just 'bsub' will work.
- Path to bkill: Where to find the bkill executable on the remote cluster. Note that if bkill is in the cluster's \$PATH, setting this to just 'bkill' will work.
- Path to bjobs: Where to find the bjobs executable on the remote cluster. Note that if bjobs is in the cluster's \$PATH, setting this to just 'bjobs' will work.

A new template, "job.lsf" is added to the list of available templates. This is the job submission script for LSF. It may take some experimentation to get jobs to submit successfully, and you may need to contact the system administrators of the cluster for assistance or information about MPI, executable locations, etc. Perhaps the easiest method to find the correct LSF script is to run some trial submissions by hand, and then replace the structure/search specific information with the appropriate keywords once a working script has been generated.

For more details on optimization schemes, see [Optimization Schemes](#).

[Skip to next section.](#)

1.4.5 Using a Remote LoadLeveler Cluster



The image shows a 'LoadLeveler Queue Configuration' dialog box. It is divided into two sections: 'Global Queue Interface Settings' and 'Step Queue Interface Settings'. The 'Global' section includes fields for Host (rush.ccr.buffalo.edu), User (name), Working directory (Server) (/path/to/dir/), Working directory (Local) (/path/to/dir/), Description (test), Queue refresh interval (10), and three checkboxes: 'Clean remote directories when finished', 'Log error directories', and 'Automatically cancel job after' (set to 100.00 hours). The 'Step' section includes fields for Path to llsubmit (llsubmit), Path to llcancel (llcancel), and Path to llq (llq). At the bottom right are 'Cancel' and 'OK' buttons.

Global Queue Interface Settings	
Host:	rush.ccr.buffalo.edu
User:	name
Working directory (Server):	/path/to/dir/
Working directory (Local):	/path/to/dir/
Description:	test
Queue refresh interval:	10
<input type="checkbox"/> Clean remote directories when finished	
<input type="checkbox"/> Log error directories	
<input type="checkbox"/> Automatically cancel job after:	100.00 hours

Step Queue Interface Settings	
Path to llsubmit:	llsubmit
Path to llcancel:	llcancel
Path to llq:	llq

Select "LoadLeveler" from the list of Queues, and then click the "Configure..." button. The step queue interface settings are:

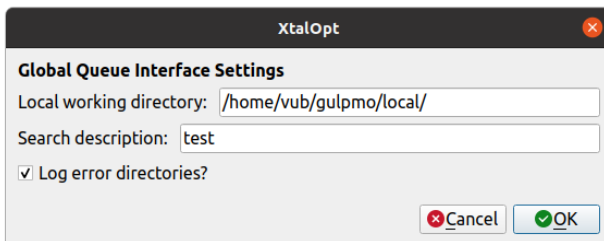
- Path to llsubmit: Where to find the llsubmit executable on the remote cluster. Note that if llsubmit is in the cluster's \$PATH, setting this to just 'llsubmit' will work.
- Path to llcancel: Where to find the llcancel executable on the remote cluster. Note that if llcancel is in the cluster's \$PATH, setting this to just 'llcancel' will work.
- Path to llq: Where to find the llq executable on the remote cluster. Note that if llq is in the cluster's \$PATH, setting this to just 'llq' will work.

A new template, "job.ll" is added to the list of available templates. This is the job submission script for LoadLeveler. It may take some experimentation to get jobs to submit successfully, and you may need to contact the system administrators of the cluster for assistance or information about MPI, executable locations, etc. Perhaps the easiest method to find the correct LoadLeveler script is to run some trial submissions by hand, and then replace the structure/search specific information with the appropriate keywords once a working script has been generated.

For more details on optimization schemes, see [Optimization Schemes](#).

[Skip to next section.](#)

1.4.6 Running Optimizations Locally



Select "Local" from the list of Queues, and then click the configure button. A new window will prompt for:

- Local working directory: A directory that is readable/writable by the current user on the local computer. This is where the final structures and resume files are written.
- Search description: a name to be displayed at the top of the XtalOpt window.
- Log error directories: if checked, structures that produce an error will have their directory saved within an "error<← Dirs" directory in the local working directory. The structure's directory will be named "<generation>x<id<← Number>". This setting can be useful for debugging errors. Note that a structure will overwrite its error directory if another error occurs.

If the optimizer's executable (vasp, gulp, pw.x, castep, etc) is not in your system path, you will need to specify the location of the executable by clicking the "Configure..." button next to the optimizer selection menu.

1.5 What Is Written to the Local Directory?

A directory for each structure is created at

```
[Local working directory]/<gen#>x<id#>
```

that contains input, output, and data files specific to each structure. Two additional files are also written to the local filesystem:

```
[Local working directory]/xtalopt.state
```

that includes save/resume information to continue a session that has been stopped, and

```
[Local working directory]/results.txt
```

that stores a list of all structures sorted by increasing enthalpy. The latter file is handy for offline analysis, since there is no need to open XtalOpt to find the most stable structures of a previous search. See [Reading the results.txt File](#) for more info.

In case either the "xtalopt.state" file or the "results.txt" file become corrupted, a previous copy of these files is also saved as "xtalopt.state.old" and "results.txt.old", respectively.

Note: If a user wishes to change some XtalOpt options before resuming a session, the "xtalopt.state" file may be directly edited before resuming. Note also, however, that if the "saveSuccessful" keyword in the "xtalopt.state" file is equal to "false", the "xtalopt.state.old" file will be automatically loaded instead.

As of XtalOpt 14, the file "hull.txt" will be written in the local directory, which includes the composition (atom counts) and total enthalpy of successfully optimized structures. This file can be used to plot the convex hull (most relevant for multi-element chemical systems). Further, and if the user instructs the code through the input flag "saveHullSnapshots", a copy of the "hull.txt" file will be saved in the "movie/" folder in the local directory after each new successful local optimization. These files, produced to facilitate monitoring the convex hull evolution, are named using a unique "data-time" string as "YYMMDD_HHmmSS_LLL", where YY (year), MM (month), DD (day), HH (hour), mm (minute), SS (seconds), and LLL (milliseconds).

1.5.1 Reading the results.txt File

The "results.txt" file is written in the local working directory, and it contains a lot of information that can be useful for offline analysis.

The "results.txt" file will typically look something like the following:

Rank	Struct	Formula	Compos	Index	EnthalpyAtm	Front	AbovHullAtm	SG	Status
1	3x10	O2Ti6	1:3	83	-8.544070	0	0.000000	1	Optimized
2	7x47	O2Ti2	1:1	387	-8.789620	0	0.000000	1	Optimized
3	9x15	O14Ti7	2:1	559	-8.829538	0	0.000000	2	Optimized
4	6x99	O1Ti2	1:2	572	-8.711570	0	0.000000	12	Sim(3x44)
5	4x98	O1Ti6	1:6	591	-8.276672	0	0.000000	1	Optimized
6	8x116	O3Ti2	3:2	697	-8.820659	0	0.000000	1	Optimized
7	5x99	O1Ti4	1:4	708	-8.429630	0	0.000000	1	Optimized
8	8x70	O1Ti2	1:2	561	-8.711543	1	0.000027	12	Sim(3x44)
9	7x51	O2Ti2	1:1	422	-8.789589	2	0.000031	1	Optimized

The following is a description of each column:

- Rank: The rank from the lowest distance above hull to the highest value.
- Struct: The "Tag" of the structure: generation and ID (offspring number) of the structure combined into a single entry as "generation"x"ID".
- Formula: The "full" chemical formula of the unit cell.
- Compos: The "reduced" composition (ratio of atom counts) of the structure.
- Index: The order in which structures are generated (0 is the first structure, etc).
- EnthalpyAtm: The enthalpy per atom of the structure.
- Front: The calculated Pareto front of the structure; most relevant for a multi-objective search. This is being updated as more structures are successfully optimized.
- AbovHullAtm: The distance above the convex hull of the structure. This is updated with every new successful local optimization, and serves as the primary target value for global optimization of the energetic quantity.
- SG: The space group number of the structure. This is obtained using the spglib library, and according to the user-defined precision.
- Status: The current status of the structure. If the status is "Sim(...)", the tag of the structure to which it is similar will be specified in the parentheses. When performing local optimization, the status will be "Step#" where "#" is the optimization step.

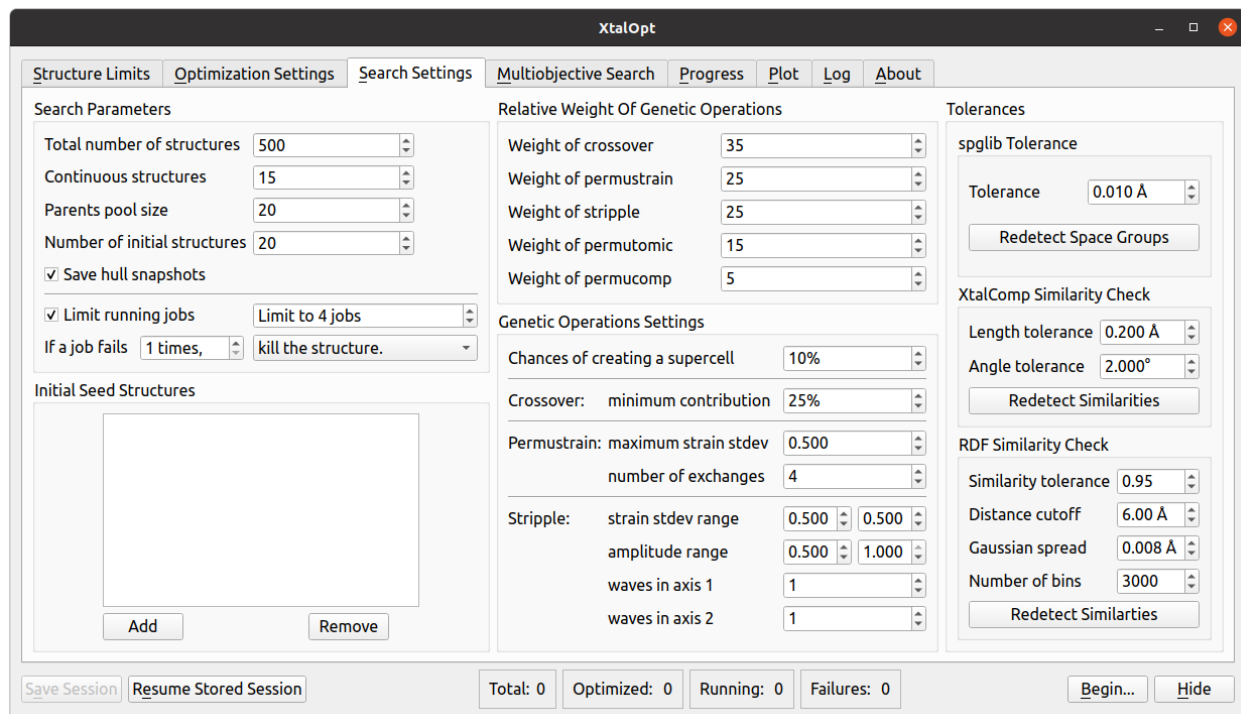
The output files for any structure can be examined by looking into its corresponding tag (i.e., <generation>x<id←Number>) directory within the local working directory (see [What Is Written to the Local Directory?](#) for more information).

It should be noted that if XtalOpt is built with the cmake flag

```
-DXTALOPT_DEBUG=ON
```

the output of the run is saved in the local working directory as the "output.log" file.

1.6 Search Settings



In the "Search Settings" tab, most of the default settings should suffice (more details [here](#)). We arbitrarily set the initial structures to 20 and the continuous structures to 15, although these may need to be adjusted based on available resources. We will not specify initial seeds, but the option to do so exists on this screen.

Note: when using RandSpG, our [tests](#) have shown that it is beneficial to create more initial structures, such as 50.

It is not necessary to limit the number of submitted jobs, unless XtalOpt is run locally. Once running the code on a cluster, the queueing system will manage the job control for us. If running locally, however, the job limit should be set no higher than the "number of available cpu cores - 1" (e.g. for a quadcore processor, allow three jobs to run simultaneously). This allows one core to remain free for the system to run.

For a run in the GUI mode, the search will pause once the "Total Number of Structures" are generated until the user either exits XtalOpt or increases this value, so that the code resumes the run with producing a new structure.

Further, in this tab the user can set the relative weight of applying the genetic operations. Generally, for fixed- and multi-composition searches only crossover, strippl, and permustrain are applicable, while for a variable-composition search all genetic operations can be used. These include two new genetic operations permutomic (randomly adding or removing an atom in the parent structure after a slight cell distortion), and permucomp (producing a new random composition in the slightly distorted parent structure's cell).

At the time of choosing a genetic operation, XtalOpt finds the percentage chance of applying a genetic operation by dividing its weight by the sum of weights of all relevant operations for the search type. Some of these operations can be fine-tuned through relevant entries found in this tab.

As of XtalOpt 14, the mitosis operation is obsolete. Instead, and with the percent chance specified by the user, the outcome of any of the genetic operations will be expanded to a supercell (within the maximum number of atoms limit),

and a randomly chosen atom in the supercell will be slightly displaced. This option, which can be set in this tab, by default has a value of 0, hence, no random supercell expansion will be performed.

The tolerances for similarity check with XtalComp are also found in this tab. They can be adjusted at any point in the run and the results.txt file will update with the correct duplicates found based upon the new tolerances. As of XtalOpt 14, another similarity check option is added: to measure the dot product of the normalized radial distribution function (RDF) of two cells and compare it against a threshold to identify similar structures (the implementation follows the methodology described [here](#)). By default, the threshold for RDF similarity check is 0, hence, XtalOpt uses XtalComp. If this threshold is set to a non-zero value in the (0,1] range (as we did here), structures with RDF dot product greater than the specified value are marked as similar. Given that a RDF dot product of 1 implies an exact match between the structures; typically a value of 0.95-0.98 for the RDF tolerance is a reasonable threshold for detecting "similar" structures in an evolutionary search.

1.7 Optimization Type and Multi-Objective Search

As of XtalOpt 14, the code can use both a basic scalar fitness function and Pareto optimization for global optimization. The basic fitness function involves a weighted sum of normalized values of optimization targets, while the Pareto optimization uses the [NSGA-II algorithm](#) to evolve the population with selecting new parent structures through a binary tournament selection based on the ranks and crowding distances of individual members of population.

XtalOpt is able to perform the multi-objective search by simultaneously optimizing two or more properties of a crystal structure. This typically includes a structure's energy or enthalpy, along with a set of user-defined objectives. These objectives include hardness (e.g., calculated using the sheer modulus obtained from AFLOW-ML and the Teter model), symmetry (space group number), volume per atom, superconducting critical temperature, or any property that can be represented by a single numerical value.

1.7.1 Specifying Objectives

As for the optimization targets (objectives), in principle, the user can define an arbitrary number of objectives for a multi-objective search. For any property that the user wants to use as an objective in the multi-objective search, there should be an executable user-provided script or code that (i) calculates that property from the output of a regular XtalOpt search, and (ii) produces an output file with a single number (integer or double) as the value of the desired objective. This output file will be read in by the XtalOpt code and will be used for determining a structure's fitness for procreation or for filtering the parent pool.

Essentially, after each local relaxation that can be performed with any of the total energy calculation methods available in XtalOpt, the code generates a structure file named output.POSCAR (this file uses the VASP format). The user-provided script should be able to read/use this file (or convert it to another structural data format if needed) to perform the intended calculations and produce the output file.

The user-provided script should be an executable script and be available in a path accessible either in the local path (for a local run) or on the cluster access path (for a remote queue). XtalOpt will call this script automatically and will read and use its output for the multi-objective optimization. The output file generated by the user-provided script should be a text file with the calculated value of the corresponding objective written as the first entry of the first line of the file.

The script can be simply a sequence of commands that use the output.POSCAR file and call some program to produce a result; or can be a bash script that actually generates a cluster job file and submits the job to the computational cluster on its own if the intended calculations are computationally demanding (see [Examples of User-defined Scripts](#) for an example of such a script).

In case the particular calculations in the script need more input data (e.g., ab initio charge density, density of states, etc.), the user-provided script should include introductory steps to generate them, or alternatively the user can add appropriate entries to the XtalOpt job template used in the structure search to produce these data.

In general, XtalOpt considers the output file produced by the script or code to be correct and contain a valid value only if the "first entry" in the "first line" of the file is a numerical value. Otherwise, e.g., the file is not produced, is empty, or its first entry is not a legitimate numerical value, the calculation will be marked as failed, and the structure will not be considered as a candidate to enter the breeding pool.

In order to invoke the multi-objective functionality, for every desired objective, the user should provide the following information to the XtalOpt code:

- **objective type:** instruction for the code on how to use the result of an objective calculation in determining the fitness function. XtalOpt can minimize or maximize an objective, or use the results of objective calculations for filtering the parents' pool.
- **path to the user-defined script:** the full path to the script corresponding to the introduced objective. The script will be automatically run by the XtalOpt code after local relaxation, and calculates the desired property.
- **script's output filename:** the name of the file generated by the script that contains the result of the calculation for the corresponding objective.
- **optimization weight:** a number between 0.0 and 1.0 that will be used as the weight for the corresponding objective in calculating the fitness function. The total weight of all objectives should not exceed 1.0, and the weight for optimization of the enthalpy will be calculated by XtalOpt as: "1.0 - total weight of the objectives". Note that while weights are not applicable in the case of "Pareto" optimization, they must be always specified, as the user has the option to change the global optimization type from "Basic" to "Pareto" and vice versa, during the run.

The user can find the optimization type and relevant settings in the "Multiobjective Search" tab, and can specify the desired objectives there (the corresponding settings in the CLI mode are also available, as described in [Multi-Objective Runs in the XtalOpt CLI](#)).

1.7.2 Optimization Type

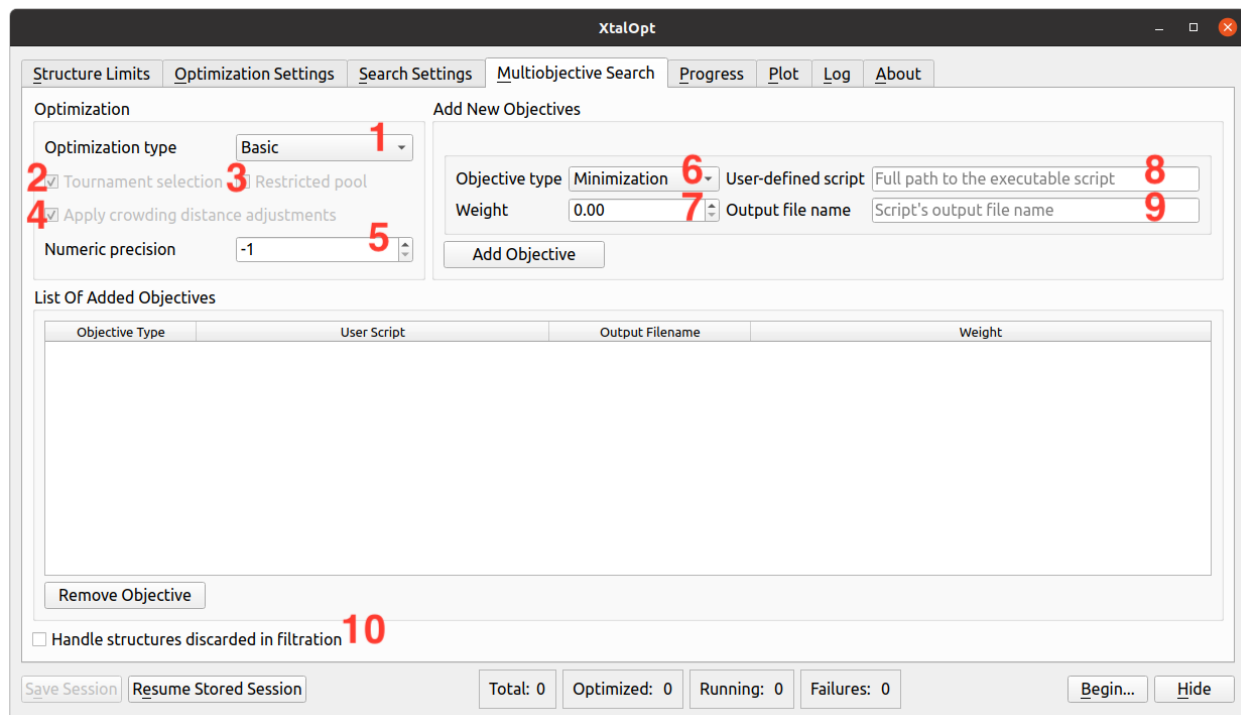
XtalOpt can calculate the probability of selecting a candidate structure for breeding by using either the "basic scalar fitness function" or through "Pareto optimization". These schemes work for both single- and multi-objective searches. By default, XtalOpt uses the "Basic" optimization type. However, the user can overwrite this behavior by explicitly setting the optimization type to "Pareto" in the "Multiobjective Search" tab, in the "Optimization" section.

In Pareto optimization, the code performs a standard non-dominated sorting, and calculates the scaled crowding distances of candidate structures in each front. This information is then used to perform a tournament selection, where the best candidate is selected from a pair of randomly selected structures. By default, XtalOpt uses the crowding distance, and selection of the pair of structures is made from the entire breeding pool.

The user can change these settings by restricting the pool for selection of the pair of structures, and disable the use of crowding distances, by un-checking the relevant options.

Additionally, if the user un-checks the tournament selection, then XtalOpt uses the ranks (and optionally crowding distances) to calculate a Pareto-based scalar fitness and uses that to select a parent for genetic operations.

1.7.3 Multi-Objective Runs in the XtalOpt GUI



In the "Multiobjective Search" tab (introduced first in XtalOpt 13), the user can specify all entries relevant to the optimization type, as well as the multi-objective run (described above) can be entered in the corresponding fields by:

1. choosing the optimization type from a drop-down menu, i.e., "Basic" to let the code use the generalized fitness function, and "Pareto" for performing the Pareto optimization.
2. if the user chooses the Pareto optimization, the parent selection can be made through tournament selection, or (if un-checked) using the Pareto-based scalar fitness.
3. for Pareto optimization with tournament selection, the user can restrict the selection of the pair of structures to the "top" candidates in the pool (i.e., as many as the pool size with best ranks and crowding distances).
4. upon Pareto optimization selection, the user can choose if crowding distance should be applied or not,
5. by default, XtalOpt uses the objective(s) value as is; however, the user can specify the number of decimal digits to be retained as an integer (default value of -1 corresponds to non-rounding).
6. choosing the objective type from a drop-down menu, i.e., "Minimization", "Maximization", and "Filtration" (see [Filtering Structures: Constrained Search](#)).
7. entering or setting the weight.
8. entering the "full path" to the external code (or script) that calculates the objective.
9. entering the output file name that the external code generates with the objective value (to be saved in the structure's working directory).
10. specifying whether a structure discarded by a filtration feature requires further handling or not (see [Filtering Structures: Constrained Search](#) for more details).

After setting each objective's entries, the selected objective can be added to the list of objectives for the run.

1.7.4 Filtering Structures: Constrained Search

The multi-objective search in the XtalOpt code can facilitate a constrained evolutionary search. Besides maximizing or minimizing a particular objective, XtalOpt allows for filtering the relaxed structures based on some property. The result of such a constraint will prevent structures deemed unsuitable from entering the parent pool, hence promoting or prohibiting the propagation of a specific genetic characteristic.

For "filtration", similar to the "minimization" and "maximization" cases, the user should provide a script that marks the structures to keep or discard based on the intended property. Structures that are marked for discarding will not be allowed in the parents' pool, although they will remain in the set of generated structures.

In the case of the filtration feature, the user can optionally instruct XtalOpt to attempt a replacement for that particular structure, according to the value of the "If a job fails" entry in the "Search Settings" tab. If these settings are equivalent to "replace with random" or "replace with offspring", the failed structure is replaced with a new structure generated randomly or by applying evolutionary operations, respectively. Otherwise, no further action is taken. If the user instructions result in replacing the failed structure with a new one, it will then be submitted for local optimization and the subsequent calculation of objectives, including the filtration objective. It should be noted that this procedure will be performed at most once for a failed structure, i.e., no more than one replacement will be attempted for a structure that is marked to be dismissed in filtration.

1.7.5 Examples of User-defined Scripts

The following is an example of a script that can be employed for a multi-objective XtalOpt run. We choose a simple example, wherein the goal is to minimize the enthalpy, while simultaneously maximizing a structure's space group number calculated from the VASP format output.POSCAR structure file. This can be done via a simple Python script, e.g., /path/spg.py, where the **Atomic Simulation Environment** is utilized to resolve the space group of the structure as,

```
import ase.io as io
from ase import Atoms
from ase.spacegroup import get_spacegroup
s=io.read('output.POSCAR', index ='-1', format='vasp')
print(get_spacegroup(s, symprec=1e-3).no)
```

The output of this short Python code can be used via a simple executable bash script, e.g., /path/spg.sh,

```
#!/bin/bash
/path/python /path/spg.py > spg.dat
```

to produce a spg.dat file containing the space group number of the structure, which is readable by XtalOpt for this objective.

Alternatively, if the user desires to submit this calculation to a computational cluster, the executable script /path/spg.queue.sh in its most basic form can be written as:

```
#!/bin/bash
cat > fspg.slurm << EOF
#!/bin/bash
#SBATCH --nodes=1 --ntasks-per-node=1
#SBATCH --job-name=fspg
#SBATCH --output=fspg.out --error=fspg.err
#SBATCH --time=00:05:00
#SBATCH --cluster=slurm
##### main task: calculating the objective
/path/python /path/spg.py > spg.dat
#####
EOF
sbatch fspg.slurm
```

This particular executable script writes a job submission script for the slurm cluster to disk (fspg.slurm file, which includes everything between the lines containing the "EOF" keywords) and then submits this job (with "sbatch fspg.slurm") to the cluster. The job submission script (fspg.slurm file) includes an introductory part (job- and cluster-related settings) and the core task, just like a usual job submission script. It contains the previous simple script for calculating the space group number (which is enclosed between "#####" comment lines for clarity). The latter script can be employed in conjunction with the multi-objective search to optimize the space group number just as in the previous example, only, this time each calculation is submitted to the cluster instead of running through a simple executable bash script.

As of XtalOpt14, the explicit support for the AFLOW-ML hardness optimization is removed. The user, however, can optimize AFLOW-ML hardness (as well as any other properties obtained from this platform) using a proper objective script, similar to any other objective.

In general, various properties such as structure-dependent band-gaps (or metal/insulator classification), bulk and shear moduli, (constant volume or pressure) heat capacity, Debye temperature, thermal expansion coefficient, and unit cell energy can be **obtained** from the AFLOW-ML interface. The user can utilize the following script to retrieve these properties from AFLOW-ML (and, e.g., further process them to calculate the desired hardness measure):

```
#!/usr/bin/python3
import json, sys, os
from time import sleep
from urllib.parse import urlencode
from urllib.request import urlopen
from urllib.request import Request
from urllib.error import HTTPError
SERVER="http://aflow.org"
API="/API/aflow-ml"
MODEL="plmf"
poscar=open('POSCAR', 'r').read()
encoded_data = urlencode({'file': poscar}).encode('utf-8')
url = SERVER + API + "/" + MODEL + "/prediction"
request_task = Request(url, encoded_data)
task = urlopen(request_task).read()
task_json = json.loads(task.decode('utf-8'))
results_endpoint = task_json["results_endpoint"]
results_url = SERVER + API + results_endpoint
incomplete = True
while incomplete:
    request_results = Request(results_url)
    results = urlopen(request_results).read()
    results_json = json.loads(results)
    if results_json["status"] == 'PENDING':
        sleep(10)
        continue
    elif results_json["status"] == 'STARTED':
        sleep(10)
        continue
    elif results_json["status"] == 'FAILURE':
        print("Error: prediction failure")
        incomplete = False
    elif results_json["status"] == 'SUCCESS':
        print("Successful prediction")
        print(results_json)
        incomplete = False
```

1.8 "Begin"

XtalOpt - test @ vortex.ccr.buffalo.edu

Structure Limits Optimization Settings Search Settings Multiobjective Search **Progress** Plot Log About

	Structure	Formula	Job ID	Status	Time Elapsed	Enthalpy/Atom	Above Hull/Atom	Front	Volume/Atom	Space Group	Ancestry
1	1x1	O12Ti6	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	N/A	N/A	19.53	104: P4nc	RandSpg Init: 104 ...
2	1x2	O12Ti6	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	N/A	N/A	7.54	47: Pmmm	RandSpg Init: 47 ...
3	1x3	O10Ti5	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	N/A	N/A	17.18	168: P6	RandSpg Init: 168 ...
4	1x4	O4Ti4	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	N/A	N/A	26.39	26: Pmc2_1	RandSpg Init: 26 ...
5	1x5	O32Ti16	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	N/A	N/A	18.21	2: P-1	RandSpg Init: 2 ...
6	1x6	O8Ti4	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	N/A	N/A	10.99	191: P6/mmm	RandSpg Init: 191 ...
7	1x7	O20Ti10	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	N/A	N/A	16.08	194: P6_3/mmc	RandSpg Init: 194 ...
8	1x8	O32Ti16	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	N/A	N/A	15.36	87: I4/m	RandSpg Init: 87 ...
9	1x9	O20Ti10	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	N/A	N/A	18.01	131: P4_2/mmc	RandSpg Init: 131 ...
10	1x10	O4Ti2	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	N/A	N/A	7.05	132: P4_2/mcm	RandSpg Init: 132 ...
11	1x11	O4Ti4	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	N/A	N/A	12.44	118: P-4n2	RandSpg Init: 118 ...
12	1x12	O6Ti3	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	N/A	N/A	6.94	123: P4/mmm	RandSpg Init: 123 ...
13	1x13	O6Ti3	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	N/A	N/A	12.28	183: P6mm	RandSpg Init: 183 ...
14	1x14	O16Ti8	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	N/A	N/A	10.57	184: P6cc	RandSpg Init: 184 ...
15	1x15	O22Ti11	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	N/A	N/A	11.93	162: P-31m	RandSpg Init: 162 ...
16	1x16	O24Ti12	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	N/A	N/A	7.12	103: P4cc	RandSpg Init: 103 ...
17	1x17	O32Ti16	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	N/A	N/A	16.05	80: I4_1	RandSpg Init: 80 ...
18	1x18	O30Ti15	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	N/A	N/A	6.70	75: P4	RandSpg Init: 75 ...
19	1x19	O20Ti10	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	N/A	N/A	8.19	48: Pnnn	RandSpg Init: 48 ...
20	1x20	O8Ti4	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	N/A	N/A	17.92	69: Fmmm	RandSpg Init: 69 ...

Refresh Refresh period: 1 seconds ☐ Verbose output log file Remove Extra Files Refresh Hulls Refresh All

Save Session Resume Stored Session Total: 0 Optimized: 0 Running: 0 Failures: 0 Begin... Hide

Figure 1 The "Progress" tab immediately after starting a search

XtalOpt has everything it needs to start its search at this point; click the "Begin" button in the lower right corner of the application to tell it to start the search algorithm. A progress bar appears as the random first generation is created. Switch to the "Progress" tab and 20 entries will appear, all with a status of "Waiting for Optimization". Click "Refresh" on this tab to begin the local optimizations. From here, XtalOpt will continue to run without user input, starting new optimizations and generating new structures until it is stopped by the user.

1.9 Monitor Progress

XtalOpt - test @ vortex.ccr.buffalo.edu

Structure Limits

Optimization Settings

Search Settings

Multiobjective Search

Progress

Plot

Log

About

Structure	Formula	Job ID	Status	Time Elapsed	Enthalpy/Atom	Above Hull/Atom	Front	Volume/Atom	Space Group	Ancestry	
380	9x9	O28Ti4	N/A	Optimized	0:00:02	-0.115366	0.018882	5	21.43	1: P1	Stripple: 8x6 ...
381	5x67	O5Ti1	N/A	Optimized	0:00:01	-0.118280	0.020119	1	20.59	8: Cm	Crossover: 4x35...
382	9x10	O7Ti7	N/A	Optimized	0:00:01	-0.132104	0.039501	10	15.06	1: P1	Crossover: 8x15...
383	5x68	O6Ti1	N/A	Optimized	0:00:01	-0.122301	0.013726	0	20.25	12: C2/m	Stripple: 4x67 ...
384	6x65	O22Ti31	N/A	Killed	0:00:03	N/A	N/A	N/A	19.99	1: P1	Crossover: 5x50...
385	10x4	O19Ti1	N/A	Optimized	0:00:01	-0.111517	0.015260	1	23.19	8: Cm	Permustrain: 9x...
386	7x38	O10Ti1	N/A	Optimized	0:00:01	-0.114735	0.016117	3	21.98	1: P1	Crossover: 4x53...
387	3x52	O17Ti9	N/A	Optimized	0:00:01	-0.124766	0.031513	5	17.66	1: P1	Permucomp: 2x...
388	8x19	O12Ti4	N/A	Optimized	0:00:02	-0.118873	0.027827	9	19.42	1: P1	Crossover: 7x37...
389	4x71	O9Ti2	N/A	Optimized	0:00:02	-0.111930	0.027979	11	20.82	5: C2	Crossover: 2x32...
390	7x39	O23Ti8	N/A	Optimized	0:00:05	-0.119784	0.027720	8	19.25	1: P1	Crossover: 5x7 ...
391	6x66	O7Ti1	N/A	Optimized	0:00:02	-0.113691	0.020557	6	21.23	1: P1	Crossover: 5x29...
392	5x69	O9Ti4	N/A	Optimized	0:00:01	-0.122044	0.030404	8	18.26	1: P1	Stripple: 4x69 ...
393	6x67	O6Ti3	N/A	Optimized	0:00:01	-0.126710	0.028291	3	17.69	8: Cm	Permutomic: 5x...
394	8x20	O6Ti18	N/A	Optimized	0:00:04	-0.164624	0.031886	1	11.34	1: P1	Stripple: 7x14 ...
395	4x72	O15Ti4	N/A	Optimized	0:00:02	-0.116156	0.026612	7	20.04	6: Pm	Permustrain: 3x...
396	2x43	O11Ti12	N/A	Optimized	0:00:02	-0.135291	0.038480	8	14.62	8: Cm	Permucomp: 1x...
397	6x68	O6Ti1	N/A	Similar to 4x44	0:00:02	-0.117117	0.018910	2	20.70	12: C2/m	Crossover: 5x29...
398	5x70	O1Ti14	N/A	Calculating objectives...	0:00:02	-0.183543	N/A	N/A	8.67	12: C2/m	Permucomp: 4x...
399	8x21	O18Ti32	N/A	Killed	0:00:03	N/A	N/A	N/A	26.57	1: P1	Supercell[2]-...
400	5x71	O17Ti4	N/A	Calculating objectives...	0:00:02	-0.119279	N/A	N/A	20.02	8: Cm	Permustrain: 4x...
401	6x69	O50Ti30	13035	Running (Opt Step 1 of 1, 0 failures)	0:00:01	N/A	N/A	N/A	17.04	1: P1	Permustrain: 5x...
402	2x44	O8Ti4	N/A	Checking status...	0:00:01	-0.125399	N/A	N/A	17.67	8: Cm	Permustrain: 1x...

Refresh

Refresh period: 1 seconds

☐ Verbose output log file

Remove Extra Files

Refresh Hulls

Refresh All

Save Session

Resume Stored Session

Total: 402

Optimized: 358

Running: 4

Failures: 17

Begin...

Hide

Figure 2 The "Progress" tab mid-run

As XtalOpt performs the search, the progress table continuously updates, providing information about each structure. We see individuals in various stages of completion: most are optimized (in blue), structure 6x68 has been automatically marked as a duplicate (dark green) of structure 4x44 and removed from the breeding pool, structure 6x69 is currently undergoing a local optimization (light green), status of the optimization step for structure 2x44 is being examined (light blue), while objectives are being calculated for structure 5x70 (yellow).

Other useful information is displayed about each structure, such as the time spent in optimization, the optimized enthalpy, the cell volume, spacegroup, and each structure's ancestry (i.e., parent(s) and parameters for the genetic operator that generated it). A status bar on the bottom of the window shows the number of structures that are optimized, running, and failing at any given time. This information is visible regardless of which tab is currently being viewed.

In the case of a multi-objective run (XtalOpt version 13.0 and newer), in the "Progress tab", the status of a structure that has successfully finished local relaxations first changes to "Calculating objectives...", after which the status changes to "Optimized" if the objective calculations were successful, "ObjectiveDismiss" if the structure was discarded by a filtration feature, or "ObjectiveFail" if the objective calculations failed.

An additional feature of the progress table is the ability to immediately visualize any of the structures in the Avogadro2 main window (assuming Avogadro2 is open with an Avogadro2 RPC server running – more info can be found in the [View Crystals in Avogadro2](#)) – simply clicking on a row in this table will display the three-dimensional structure in Avogadro2, where it can be visualized, modified, or exported. If the user would like to add a bit of "intelligent design" to the evolutionary process, a structure can be modified and then resubmitted using a context (right-click) menu from the progress table. The context menu provides tools to (un)kill a structure, resubmit for local optimization at an arbitrary optimization step, replace a problematic structure with a new, random individual, or even generate a [simulated XRD pattern](#) for the crystal.

Three additional buttons found near the "Refresh All" button in this tab are also available. The "Remove Extra Files" button is used for VASP runs. It removes any extraneous files in each local subdirectory in order to reduce disk usage. Files kept are structure.state, POTCAR, CONTCAR, OSZICAR, job.slurm and OUTCAR. Finally, the "Rank All" button ranks all currently optimized structures and exports them to a subdirectory (Ranked) in two forms (depending on the optimizer): POSCAR/.got and .cif. Each can be found in separate directories. (Only works for GULP and VASP runs currently). Further, while various entries in the progress table are being updated automatically, a "Refresh Hulls" button is added to the tab to force refreshing the distance above the convex hull and Pareto front information, in case there is a delay in the automatic update.

To alter a run manually, right-clicking on any of the structures in the Progress tab will bring up a menu of options. These options include: Killing a structure, restarting a structure, replacing a structure, injecting a seed structure, etc. All of these can be done mid-run.

Also, in the GUI mode, one can instruct the code to produce more debug-like information in the "output.log" file through the "Verbose output log file" check box (activated only if the code is built such that the "output.log" file is created).

1.9.1 View Trends

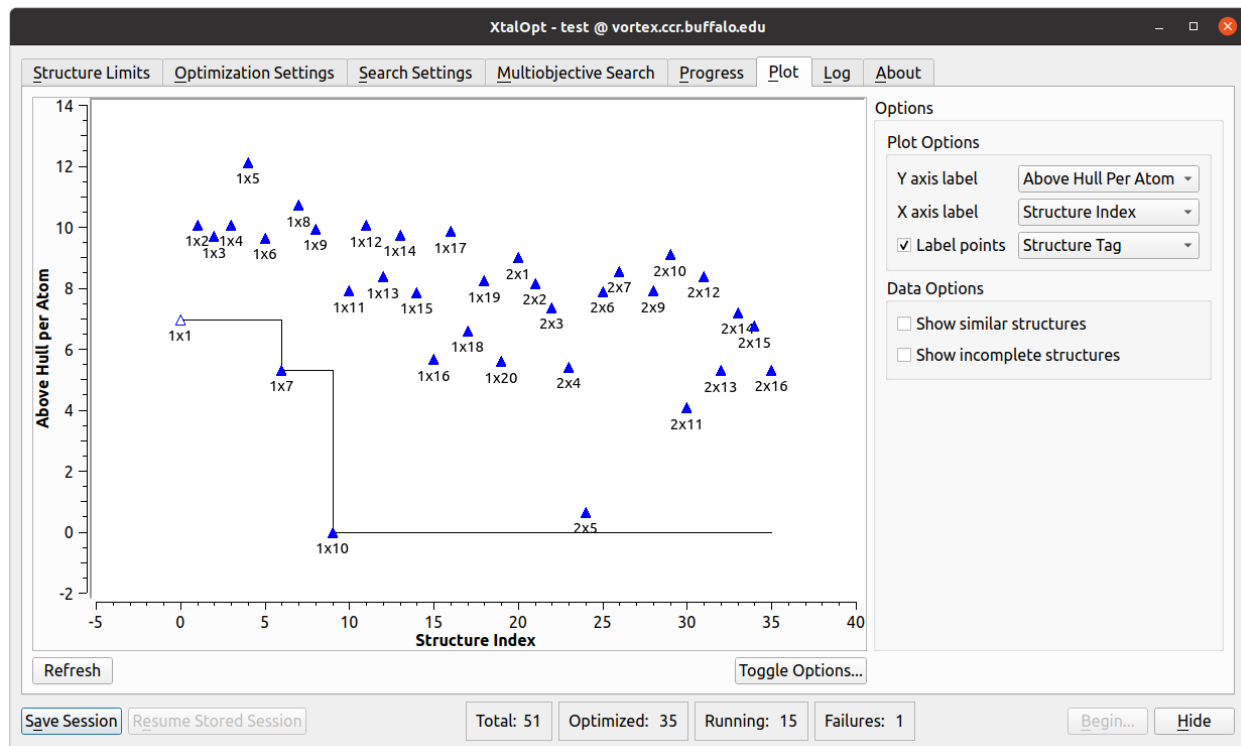


Figure 3 The "Plot" tab mid-run displaying distance above hull vs. structure index. Each structure is labeled with its Structure ID.

Another visualization and analysis tool available during the search is the interactive plot. The plot is capable of investigating trends in the search by plotting a point for each individual using structure index, generation number, enthalpy, energy, PV enthalpy term, lattice parameters, cell volume, or the objectives' values (in a multi-objective run) on either axis. This powerful feature allows the user to visualize complex relationships present in the generated structures. E.g., a plot of enthalpy vs. structure number provides an overview of the search's progress. Or, recalling that $H = U + PV$, plotting enthalpy vs. PV enthalpy term or energy lends insight into whether the enthalpy (H) is dominated by atomic interactions (U)

or cell parameters (PV). Further information is available by labeling the points with the individual's spacegroup number, Hermann Mauguin spacegroup symbol, enthalpy, energy, PV term, volume, generation, or index number.

A particularly useful plot is that of enthalpy vs. cell volume. With such a view, we can see a general trend that enthalpy increases with volume (the effect is much more pronounced for systems at higher pressures), and also that below a certain volume, enthalpy rises sharply. There will typically be a cluster of very low enthalpy structures that have a relatively small volume. Armed with this data, we can update the starting volume on the "Structure Limits" tab mid-run to reflect this new piece of information that the search has provided for us. Many of the other parameters governing structure generation and algorithm specifics can be similarly modified during a search without the need to restart the algorithm. Further, and in a variable-composition search, the trend view option to plot the distance above hull vs. structure index is a quick shortcut to identify the thermodynamically stable phases that are discovered in the search.

The plot is also interactive; zooming and panning are possible using simple mouse controls. Clicking on a structure's point on the plot will load it in the main Avogadro2 window (assuming Avogadro2 is open with an Avogadro2 RPC server running - more info in the [View Crystals in Avogadro2](#)), allowing all the same functionality as described above in [Monitor Progress](#).

In a multi-objective run, among the options to view the trends, the user can find the specified objectives, and use them to monitor the trends of various objective values in the generated structures' pool.

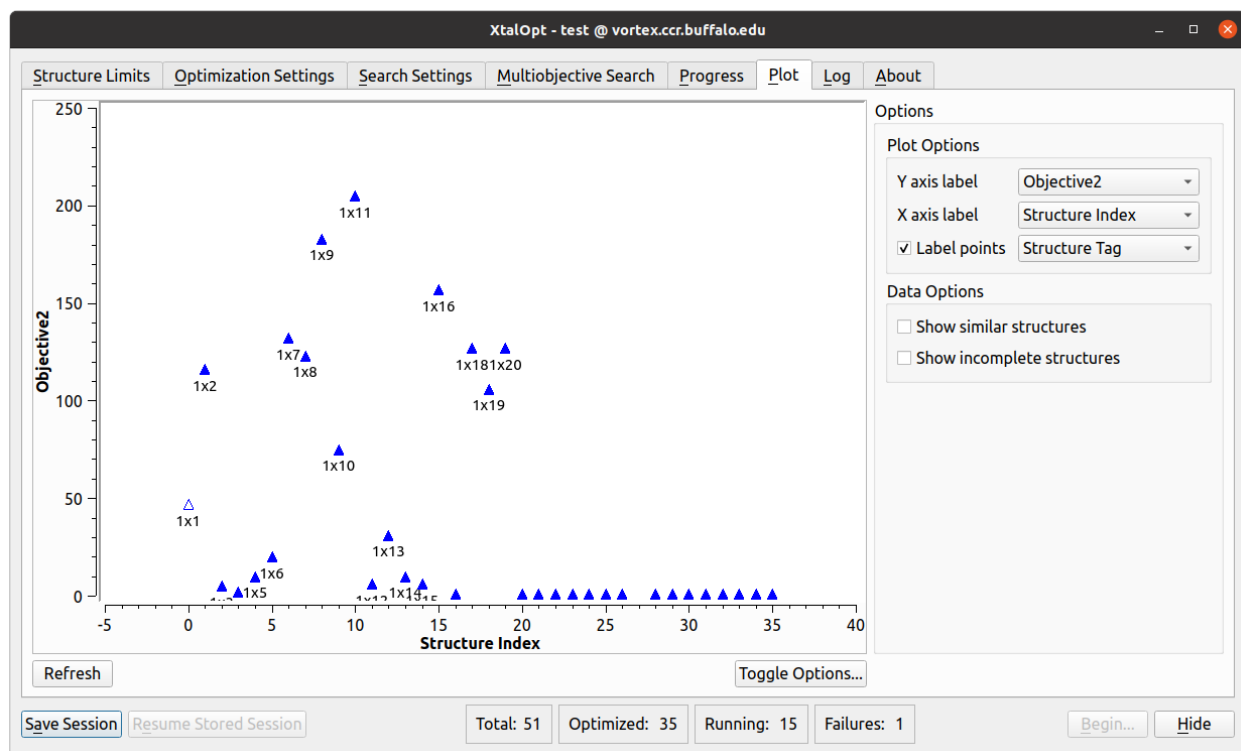
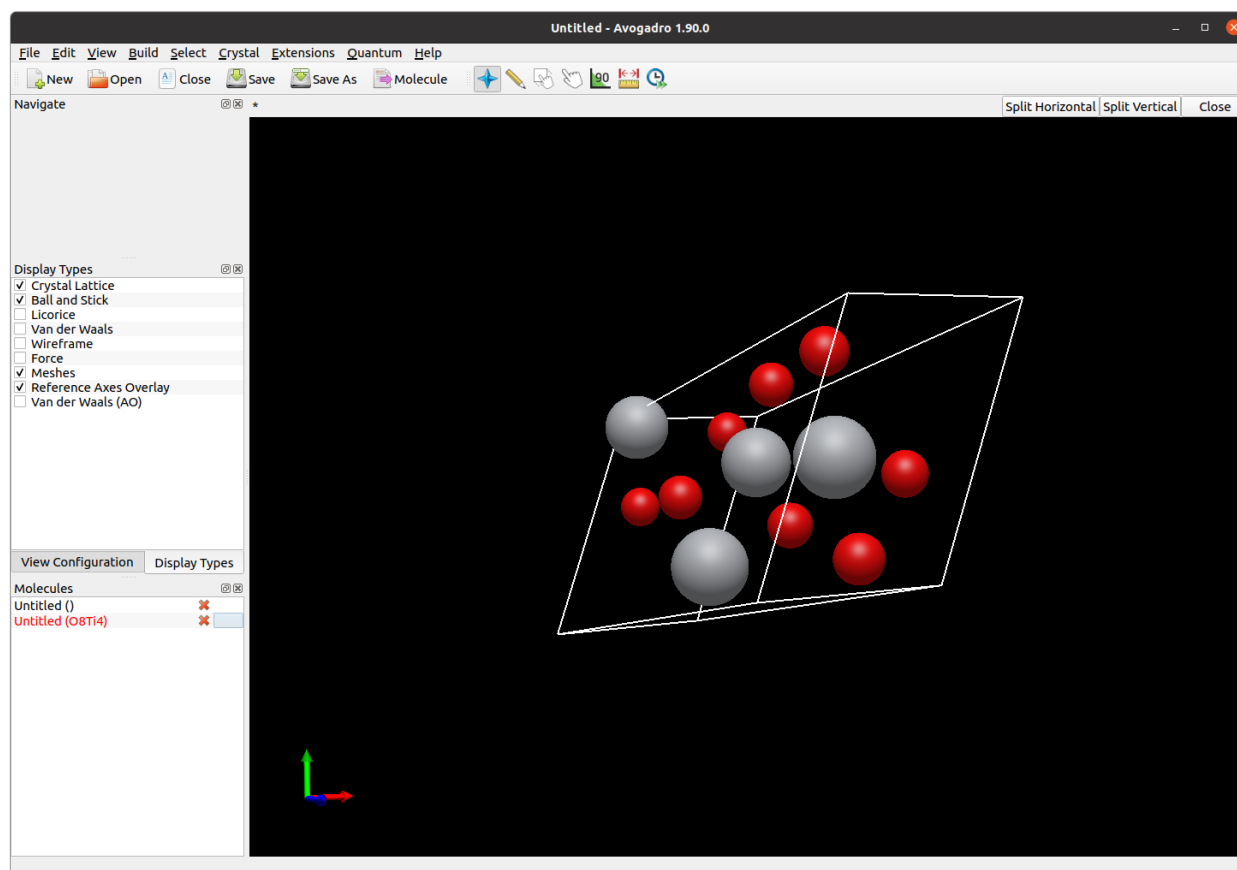


Figure 4 The plot for value of the second objective vs. structure index.

1.9.2 View Crystals in Avogadro2



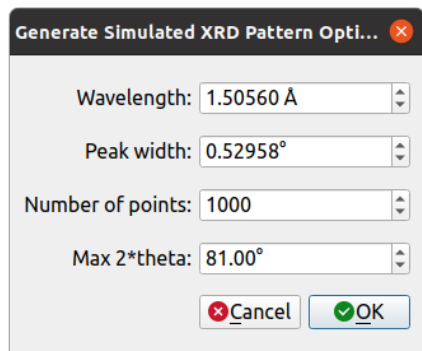
As of release 11, while an XtalOpt window is open, the user may easily view the crystals in Avogadro2. As long as an Avogadro2 window is open and an Avogadro2 Remote Procedure Call (RPC) server is running, when a user selects crystals in XtalOpt (either via the plot tab or the progress tab), the structure in the Avogadro2 window will automatically display the crystal the user selected. This allows for quick and easy visualization when analyzing a run.

1.9.3 Plot a Simulated XRD Pattern

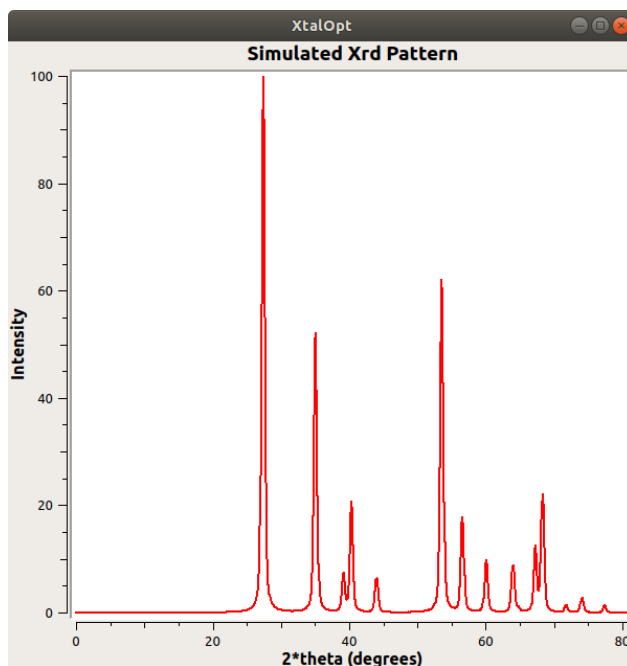
As of release 12, a simulated XRD pattern can be generated for any crystal using [Objcryst](#). A user must simply right click on any entry in the progress tab, and then click "Plot Simulated XRD Pattern" (as shown in the image below).

186	3x45	O8Ti4	N/A	Optimized	0:00:01	-0.124179
187	5x28	O8Ti15	N/A	Optimized		8946
188	4x37	O12Ti16	N/A	Calculating o		9141
189	4x38	O9Ti5	N/A	Calculating o		4328
190	4x39	O10Ti11	N/A	Checking stat		5281
191	7x8	O54Ti26	13968	Running (Opt		
192	6x18	O6Ti18	14008	Running (Opt		
193	5x29	O7Ti2	14009	Starting upda		
194	5x30	O8Ti5	N/A	Waiting for O		
195	4x40	O30Ti36	N/A	Waiting for O		
196	5x31	O7Ti1	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A

A dialog box will appear that allows a user to select several options for the XRD pattern. All of these options have tooltips that describe their function. The options include "Wavelength" (the wavelength of the x-ray), "Peak width" (which broadens the peaks to help simulate temperature), "Number of points" (to increase smoothness), and "Max 2*theta" (the max value to be displayed on the x-axis).



Once these options have been chosen, the user may simply click "OK" to generate the plot. A plot will appear similar to the one shown below.



1.10 Command Line Interface

As of release 11, there is a command line interface available within XtalOpt. This allows users to choose settings and run searches without the use of the GUI.

When performing a CLI run, an input file for XtalOpt is required, from which XtalOpt reads in the run settings (see the [List of the input flags for the CLI setting file](#) for a full list and brief description of the flags, besides the above-mentioned "--help" argument). With access to an XtalOpt binary, one can also see all of the XtalOpt CLI options by running

"xtalopt --help" from a terminal. This prints all options and a description of each one. An example that contains detailed explanations of various options can be found [here](#).

In general, all the options described in the GUI have a corresponding input for the CLI mode. For example, the custom user keywords can be introduced in the input file as:

```
user1 = ...
```

or, seed structures can be added to the run using the flag:

```
seedStructures = [path/seed1], [path/seed2], ...
```

One difference between the GUI and CLI runs is that template files in the CLI need to be created for the different templates for queue interfaces and optimization schemes. These templates use the same %keywords% that are used in the GUI, and an example of a GULP template can be found [here](#).

A CLI run is begun by entering into the terminal "xtalopt --cli". By default, XtalOpt will search for an "xtalopt.in" file in the current directory to use for the input file and read in the search parameters. If a different input file is desired, the user may explicitly select an input file with "--input-file [file]".

When a CLI run first begins, all of the options set by the user and XtalOpt's resulting settings will be printed to the terminal (and to an "settings.log" file in the local working directory). The user should glance over these printed options to ensure that they are set correctly. While a CLI run is in progress, the user is still able to update settings via a file in their local working directory called "cli-runtime-options.txt". The settings in the "cli-runtime-options.txt" file are read every iteration of the event loop. Thus, if a setting is changed in that file, it will quickly be updated in the program.

In addition, while the CLI run is in progress, job submissions, completions, and errors will be reported in the terminal. If one wishes to end the CLI run at any time, they just need to use "ctrl-C" or whatever their process interruption command is.

As of version 13.0 of XtalOpt, and for a run in the CLI mode, the code can be instructed to exit after producing the specified number of structures by adding the following flag to the input file:

```
softExit = true # default is false
```

or by setting its value to true in the run-time setting file cli-runtime-options.txt during the run. With this flag set to true, the code quits after all running (and pending) jobs are finished and the output files are updated.

On the other hand, and at any moment during a run, the user can force the XtalOpt process (hence, the run) to quit immediately by adding the following line to the run-time settings, i.e., the cli-runtime-options.txt file,

```
hardExit = true
```

It should be noted that (1) the hardExit flag terminates the XtalOpt running process regardless of any running or pending jobs and without updating the output files, and (2) this is only a run-time option and the presence of the flag in the input file is ignored by XtalOpt. One is also able to resume a run via the CLI if they type in "xtalopt --resume --dir <path/to/resume/dir>". This will check to see if an "xtalopt.state" file is in the specified directory, and if it is, XtalOpt will attempt to resume the run. Alternatively, XtalOpt can resume a CLI run in the GUI mode if that is desired instead: simply start up the GUI and resume the run using the "xtalopt.state" file as one would normally do for resuming a run started in the GUI mode.

While a CLI run is in progress (or after any run is finished), the user may view an overview of important information about the run results via the "results.txt" file in the local working directory (see [Reading the results.txt File](#) for more info). They may also generate a plot with the "xtalopt --plot --dir <path/to/dir>" command. This will immediately display a plot using the same code as that used to generate the plot tab in a GUI run. Similar to a GUI run, plot axes and other options may be changed. In addition, if Avogadro2 is open in the background and an Avogadro2 RPC server is running, XtalOpt will still also set the structure in the Avogadro2 view to the crystals the user selects.

1.10.1 Multi-Objective Runs in the XtalOpt CLI

As an example of setting up a CLI run, let's take a look at initializing the multi-objective run parameters in the CLI (compare the following parameters with those described earlier for the GUI mode in the [Optimization Type and Multi-Objective Search](#) section).

Let's assume that we have already set various search parameters similar to those in the GUI mode, e.g., corresponding to the structure limit tab settings, we have set:

```
chemicalFormulas = Ti102 - Ti16032, Ti404  
referenceEnergies = Ti3 -23.35901499, O16 -78.9700885  
vcSearch = true  
maxAtoms = 80
```

In the CLI, the optimization type can be specified through the keyword "optimizationType", which can have values of "Basic" or "Pareto", with the "Basic" being the default value:

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

For the case of Pareto optimization, the parent selection mode can be specified through the following flag:

```
tournamentSelection = False # default is True
```

and if the tournament selection is chosen, the pool can be limited to the "top" candidates with:

```
restrictedPool = True # default is False
```

Further, and the use of crowding distances in Pareto optimization which is applied by default, can be disabled using:

```
crowdingDistance = False # default is True
```

while, the precision of objective values can be set through the flag:

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

where the specified integer will set the number of retained decimal digits (the default value of -1 means no rounding for the objective values).

For each user-defined objective a line should be added to the XtalOpt input file that starts with the keyword "objective". This line includes the above-mentioned information for [Specifying Objectives](#) and, generally, has the following format:

```
objective = "objective_type" "/path_to/script" "script_output_filename" "weight"
```

where, possible options for the "objective type" are "minimization", "maximization", and "filtration", are those that have been introduced above. This field is not case sensitive, and only the first three letters are important in identifying the objective type by XtalOpt (i.e., "min", "max", "har", and "fil").

In the CLI mode, the user can instruct XtalOpt to handle a structure that is marked for discarding by a "filtration" feature with adding the following line to the input file:

```
objectivesReDo = true # default is false
```

XtalOpt will remove the structure from the parents' pool or replace it with a new one according to the user's instruction specified by the jobFailAction flag in the input file.

The output of a multi-objective run in CLI can be monitored through the "results.txt" file (introduced earlier in [Reading the results.txt File](#)). In such a run, the status of each structure in the "results.txt" file will be updated during the objective calculations. The status of a structure that has successfully been locally optimized changes to "ObjCal".

Once the objectives' values are calculated, the status changes to "Optimized", "ObjDismiss", or "ObjFail" depending on whether the calculations finished successfully, the structure was discarded during filtration, or the calculations failed, respectively.

Moreover, the "results.txt" file contains an extra column of "Objective#" for each objective introduced by the user. Before and while an objective is being calculated, its value in the relevant column is the place-holder character of "-". After successful calculation of each objective, the value will be updated accordingly.

For each structure, besides the corresponding output files generated by the scripts, a summary of the objective-related info (overall status of its calculations and their value) is written to the "structure.state" file.

Finally, in any XtalOpt run in the CLI mode the "cli-runtime-options.txt" file, which includes the parameters that can be modified during the search, is produced. Among the multi-objective-related entries only the "objectives↔ ReDo" flag is output to this file, and is allowed to be changed once the run is started (as discussed earlier in [Multi-Objective Runs in the XtalOpt GUI](#)).

1.10.2 Submitting Optimization Jobs to a Queue in a Local Run

Often when lengthy evolutionary searches are performed, the XtalOpt code is executed on the cluster where the jobs are being submitted (i.e., running XtalOpt locally while submitting the jobs to a queue). As the jobs are being submitted to the cluster, a remote queue interface should be specified in the XtalOpt input (e.g., [Using a Remote SLURM Cluster](#), [Using a Remote PBS Cluster](#), etc.). This, however, requires a ssh connection to the cluster itself, which depending on the ssh configuration of the user's account might not be allowed.

For these type of runs in the CLI mode, as of version 13.0 of XtalOpt, the user can add the following flag to the input file and run the code as usual:

```
localQueue = true # default is false
```

For more details on optimization schemes, see [Optimization Schemes](#).

1.10.3 VASP "System" POTCAR

As of version 13.0, it is possible to provide only a single POTCAR file for a multi-element system. This option is especially useful when XtalOpt is interfaced with an external code that is not explicitly supported (e.g., an arbitrary optimizer, which is scripted to produce VASP format output files). This can be done in the graphical user interface (GUI) of XtalOpt by introducing a single POTCAR file:

```
%fileContents:/path/to/system/potcar%
```

It should be noted that:

1. as XtalOpt arranges the chemical elements in alphabetical order, individual POTCAR files should be combined in the same order to produce the correct results,
2. if a "system" POTCAR is introduced in the CLI mode, other entries of potcarFile flag in the XtalOpt input file will be ignored by the code.

1.10.4 List of the input flags for the CLI setting file

A full list of the input flags with their description and default values (if any) are as follows.

input flag	description	default value
chemicalFormulas	The full chemical formula of initial cells	
vcSearch	If set to True, a variable-composition search is performed	False
referenceEnergies	Reference energies for convex hull (comma-delimited list of full formula and energy)	0 for elements
maxAtoms	The maximum number of atoms in the generated unit cells	20
optimizationType	Optimization scheme: Basic or Pareto	Basic
tournamentSelection	If set to True, use tournament selection in Pareto optimization	True
restrictedPool	If set to True, restrict the tournament selection to top structures	False
crowdingDistance	If set to True, use crowding distances in Pareto optimization	True
objectivePrecision	Number of decimal places in calculating normalized objective values	-1
objective	An objective in the multi-objective evolutionary search	
saveHullSnapshot	If set to True, save snapshots of hull data in 'movie' folder	False
verboseOutput	If set to True, produce extra information in the run output	False
user1	Custom user-defined keyword	
user2	Custom user-defined keyword	
user3	Custom user-defined keyword	
user4	Custom user-defined keyword	
usingRadiiInteratomicDistanceLimit	If set to True: use the default element-specific interatomic distances	True
radiiScalingFactor	Scaling factor for minimum radius if using scaled interatomic distances	0.5
minRadius	Absolute min radius (Å) if using scaled interatomic distances	0.25
usingCustomIADs	If set to True: use the user-defined custom interatomic distances for the elements	False
customIAD	If usingCustomIADs set, this is set for each pair of elements as: "<Symbol1>, <Symbol2>, <min↔Distance>"	
checkIADPostOptimization	Check interatomic distances after relaxations	False
maxNumStructures	Maximum number of structures to be generated in the run	100
numInitial	Number of initial structures generated in the run	0
parentsPoolSize	The population size for the parent pool	20
continuousStructures	Number of structures to be kept "In Progress" after the initial generation	15

input flag	description	default value
limitRunningJobs	If set to True: limit the number of jobs by "running↔ JobLimit"	True
runningJobLimit	Maximum number of local relaxations submitted in a remote run	1
seedStructures	Comma-separated list of full path to seed structures	
hardExit	Exit the run immediately if set to True	False
softExit	Exit the run once maximum number of structures generated	False
weightCrossover	Relative weight for applying crossover genetic operation	35
weightStripple	Relative weight for applying stripple genetic operation	25
weightPermustrain	Relative weight for applying permustrain genetic operation	25
weightPermutomic	Relative weight for applying permutomic genetic operation (vcSearch only)	15
weightPermucomp	Relative weight for applying permucomp genetic operation (vcSearch only)	5
randomSuperCell	Percent chance of producing random supercell expansion from genetic operation output	0
aMax	Maximum of the "a" lattice vector length	10.0
aMin	Minimum of the "a" lattice vector length	3.0
bMax	Maximum of the "b" lattice vector length	10.0
bMin	Minimum of the "b" lattice vector length	3.0
cMax	Maximum of the "c" lattice vector length	10.0
cMin	Minimum of the "c" lattice vector length	3.0
alphaMax	Maximum of the lattice angle "alpha"	120.0
alphaMin	Minimum of the lattice angle "alpha"	60.0
betaMax	Maximum of the lattice angle "beta"	120.0
betaMin	Minimum of the lattice angle "beta"	60.0
gammaMax	Maximum of the lattice angle "gamma"	120.0
gammaMin	Minimum of the lattice angle "gamma"	60.0
maxVolume	Maximum volume of the generated unit cell in $\text{\AA}^3/\leftrightarrow$ Atom	100.0
minVolume	Minimum volume of the generated unit cell in $\text{\AA}^3/\leftrightarrow$ Atom	1.0
maxVolumeScale	Maximum scaling factor for covalent volume	0.0
minVolumeScale	Minimum scaling factor for covalent volume	0.0
elementalVolumes	Volume limits for elements as <full_cell_formula min max> in $\text{\AA}^3/\text{cell}$ units	
usingRandSpg	If set to True: use randSpg to generate initial generation	False
forcedSpgsWithRandSpg	Create initial structures of the specified list of space groups	

input flag	description	default value
usingMolecularUnits	Use the molecular units in building the first random generation (not compatible with randSpg)	False
jobFailLimit	Follow the "jobFailAction" after this many failures in local optimization of a structure	1
jobFailAction	Action to be taken after specified number of failed local optimization	replaceWithRandom
autoCancelJobAfterTime	If set to True: the still-running jobs will be canceled after "hoursForAutoCancelJob"	False
hoursForAutoCancelJob	Time limit (hours) to cancel a still-running local optimization job	100.0
optimizer	The structure optimizer interface (e.g., vasp, pwscf, etc)	
numOptimizationSteps	Number of local optimization steps to be performed	1
jobTemplates	Comma-separated list of job templates for a remote queue	
incarTemplates	Comma-separated list of incar templates for VASP optimizer	
kpointsTemplates	Comma-separated list of kpoints templates for VASP optimizer	
potcarFile	Comma-separated list of VASP potcar files for elements/system	
pwscfTemplates	Comma-separated list of templates for PWSCF optimizer	
ginTemplates	Comma-separated list of GULP input templates	
castepParamTemplates	Comma-separated list of CASTEP parameter templates	
castepCellTemplates	Comma-separated list of CASTEP cell templates	
fdfTemplates	Comma-separated list of SIESTA templates	
psfFile	Comma-separated list of SIESTA "symbol.psf" files	
queueInterface	The target queue interface for the run: "local" or a remote queue, e.g., "slurm", "pbs".	
localQueue	If set to True: submit the jobs to a queue in a local run	False
templatesDirectory	Directory to search in for the optimizer template files	current directory, ie, "."
localWorkingDirectory	The main local working directory	localWorkingDirectory
remoteWorkingDirectory	The main remote working directory	
exeLocation	Full path to the binary of the optimizer in local run	
queueRefreshInterval	Time interval (seconds) by which the queue info is obtained	10
cancelCommand	Job cancellation command for the cluster in a remote run	
statusCommand	Job status command for the cluster in a remote run	
submitCommand	Job submission command for the cluster in a remote run	
logErrorDirectories	Create a local log directory for the failed optimizations	False

input flag	description	default value
cleanRemoteDirs	Clean-up the remote working directories after relaxations are finished	False
host	Host name for the SSH connection in a remote run	
port	SSH port in a remote connection	22
user	User name for the SSH connection in a remote run	
spglibTolerance	The tolerance for the spglib symmetry detection	0.01
xtalcompToleranceAngle	Angle tolerance for detecting similar structures	2.0
xtalcompToleranceLength	Length tolerance for detecting similar structures	0.1
rdfTolerance	Threshold for similarity in RDF dot product	0
rdfCutoff	Maximum bond length considered in RDF calculation (in Å)	6.0
rdfSigma	Spread of the Gaussian in RDF calculation (in Å)	0.08
rdfNumbBins	Number of bins in RDF calculation	3000
strippleAmplitudeMax	Maximum ripple amplitude	1.0
strippleAmplitudeMin	Minimum ripple amplitude	0.5
strippleNumWavesAxis1	Number of waves in the first non-displaced direction for the ripple operator	1
strippleNumWavesAxis2	Number of waves in the second non-displaced direction for the ripple operator	1
strippleStrainStdevMax	Maximum standard deviation for the random elements in the strain matrix	0.5
strippleStrainStdevMin	Minimum standard deviation for the random elements in the strain matrix	0.5
permustrainNumExchanges	Number of separate swaps in the permutation operator	4
permustrainStrainStdevMax	Maximum standard deviation for the random elements in the strain matrix	0.5
crossoverMinContribution	Percentage for smallest contribution a parent may make to the offspring during crossover	25

Starting from XtalOpt version 14, the following input flags are obsolete.

input flag	comment
empiricalFormula	Replaced by the chemicalFormulas flag
formulaUnits	Merged into the chemicalFormulas flag
usingFormulaUnitCrossovers	Now it is always performed
formulaUnitCrossoversGen	Now it is always performed
usingOneGenePool	Now it is always performed
usingSubcellMitosis	
mitosisDivisions	
mitosisA	
mitosisB	
mitosisC	
printSubcell	
usingMitoticGrowth	
chanceOfFutureMitosis	

As mentioned in the message, these can typically be ignored if it only happens for a handful of structures. This occurs when a structure has been generated in XtalOpt, but it has not completed any geometry optimization so there are no output files from which to load the geometry. If it happens for a significant number of structures (or structures that are known to have completed at least one geometry optimization step), the output files from the optimizer may be missing or corrupt.

After resuming a session, XtalOpt will ask if you would like to continue the search or enter read-only mode. Read-only mode will not generate new structures or submit geometry optimizations.

Note

If you are considering resuming a read-only session, take a look at the results.txt file in the working directory. It contains a list of all structures, sorted by enthalpy, with additional useful information. This can save some time when trying to locate the most stable structure of an old search.

The working directories for XtalOpt are relocatable, meaning that the directory containing xtalo.state and the [gen]x[id] structure folders may be moved, tarred, zipped, etc. and still be resumed at a later time from a different location on the filesystem, or even a different computer.

3 Optimization Schemes

3.1 Overview: What Are Optimization Schemes, and Why Use Them?

3.1.1 In a Nutshell...

An optimization scheme is a series of optimization steps ("optsteps") that are to be performed in sequence on a structure. Each optimization step consists of a set of input file templates for the queuing system and optimizer to be used, and the structure is updated after each completes. So if an optimization scheme contains three optimization steps, a structure's lifecycle is:

1. Generation of initial structure
2. Perform optstep 1 on initial structure
3. Update structure from the results of optstep 1
4. Perform optstep 2 on current structure (result of optstep 1)
5. Update structure from the results of optstep 2
6. Perform optstep 3 on current structure (result of optstep 2)
7. Update structure from the results of optstep 3
8. Current structure (result of optstep 3) is either accepted into the breeding pool or discarded, depending on its enthalpy (or hardness) relative to the other optimized structures.

3.1.2 More Details

The efficiency of searching a potential energy surface for a global minimum can be significantly improved by moving each candidate structure to the nearest local minimum, i.e., performing a geometry optimization. The differences between searching with and without carrying out these local optimizations are explored in detail in [Woodley and Catlow, Comp. Mat. Sci. 45, 84 \(2009\)](#).

Why not just perform a single geometry optimization on each structure? Stochastic search techniques, such as XtalOpt, will often need to perform geometry optimizations on structures that are far from a stationary point on the potential energy surface. For example, the randomly generated structures in the first generation of an evolutionary search are often highly disordered with unrealistic atomic separations. If these structures were to be optimized in a single step with accurately small convergence criteria, it would be quite expensive. Also, it is more than likely that most of the optimizations would not finish successfully before reaching the maximum number of geometry steps allowed by the optimizer or specified in the input. A second issue is that complex structures (periodic crystals, for example) often have so many degrees of freedom that convergence in a single step is difficult from a poor starting point (consider the effect on atomic coordinates when a unit cell's translation vector is modified).

The first problem (effectively optimizing to small convergence) can be solved by implementing an optimization scheme that optimizes to successively smaller convergence cutoffs.

The second problem can be addressed by reducing the degrees of freedom in the early optsteps and only optimizing everything once each component has individually converged to a reasonable parameterization. See [Suggestions for Optimization Schemes](#) for examples.

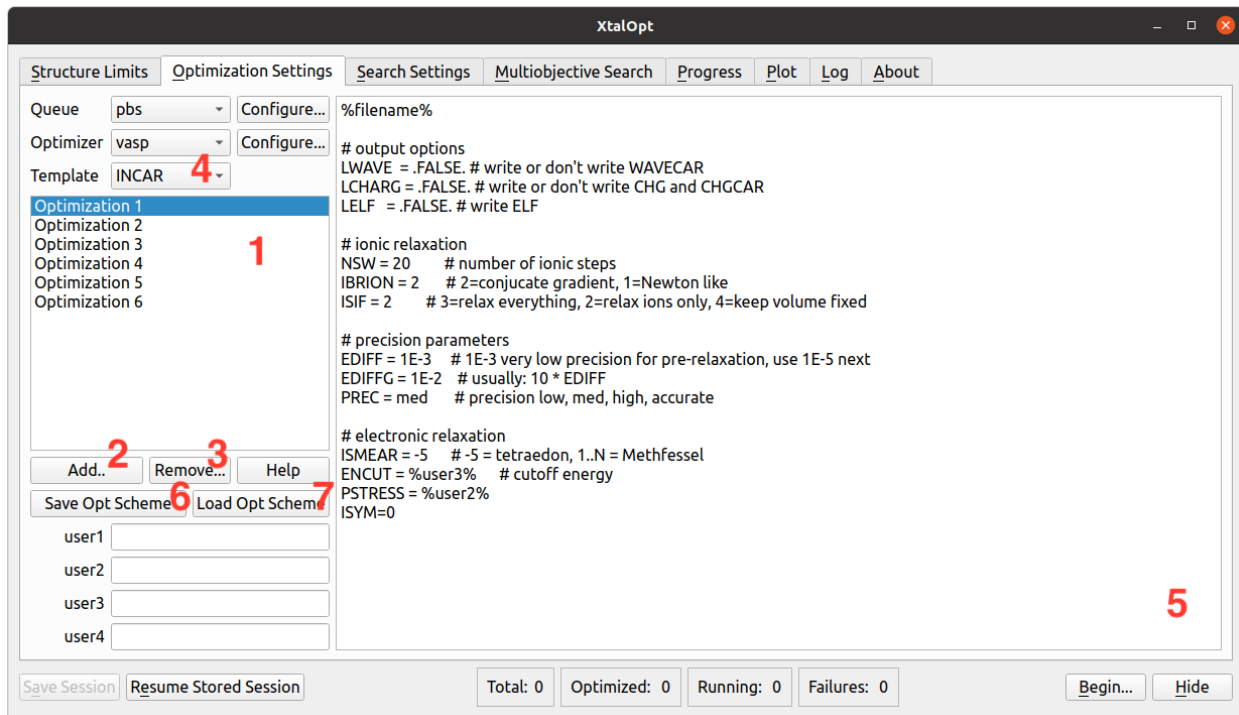
Note that before release 12, only one optimizer could be used for all of the optimization steps. However, as of release 12, different optimizers can be used for different optimization steps. Simply select the optimization step in the "Optimization Settings" tab and then choose the optimizer for that optimization step. Thus, the following could be allowed protocols:

- In optstep1 use a ML potential for a quick optimization, and in optstep2 do a DFT single point calculation to get a better prediction of the energy.

or:

- In optstep1-3 relax the structure. In optstep4 perform a short MD run to check if it is a viable local minimum.

3.2 Optimization Scheme User Interface



We will use the above screenshot as we describe the process of creating, saving, and loading optimization schemes. The numbers indicate:

1. List of optimization steps
2. Button to add new optimization step
3. Button to remove current optimization step
4. Template selection menu
5. Template editor
6. Button to save current optimization scheme to file
7. Button to load optimization scheme from file

3.2.1 Optimization Step List

This list shows the currently available optimization steps in the order that they will be performed. The optstep that is currently selected for editing is highlighted, and the editable optstep can be selected by clicking the appropriate entry.

3.2.2 Add New Optimization Step

Clicking this button will append a new optimization step to the optstep list. The new optstep's templates will be copies of the currently selected optstep's templates.

3.2.3 Remove Current Optimization Step

Click this button to delete the currently selected optimization step.

3.2.4 Select Template

This menu contains the filenames of the templates that are required by the currently selected queuing system (e.g. PBS, SGE, local...) and optimizer. The currently selected template is displayed in the template editor, and selecting a different template will update the editor.

3.2.5 Template Editor

This text editor is used to view and edit the currently selected template for the current optstep.

3.2.6 Save Scheme

This button will prompt for a location to save a .scheme file containing the current optimization step.

3.2.7 Resume Scheme

This button will prompt for an existing .scheme file to load.

3.3 How to Build an Optimization Scheme?

Creating a working scheme from scratch may take some time. We recommend checking the schemes/ directory of the source code to obtain a sample scheme for each optimizer (see [How to Load an Optimization Scheme?](#)) and verifying that they are appropriate for the system under consideration before starting a search.

If there is not an appropriate sample, the following prescription may be used to generate your own:

1. Generate a random structure of the system under consideration. This may be done by hand, or by running a search just long enough to create the first random generation and saving one of the structures.
2. Create a starting optstep with the desired convergence criteria.
3. Manually submit the optimization.
4. If the optimization fails:
 - (a) First determine why – if the maximum iterations were exceeded or the optimization was aborted due to a badly performing minimizer, try one of the ideas below. Other optimization problems are beyond the scope of this document.
 - (b) Reduce the convergence criteria of the current trial optstep.
 - (c) Remove degrees of freedom, e.g. by fixing cell parameters, atomic positions, etc.

- (d) Reduce the accuracy of the calculation in other ways (use a coarser integration grid, etc).
 - (e) Change the minimizer (e.g. tell the optimizer to use conjugate gradients rather than BFGS, etc).
 - (f) Note that not all optimizations need to converge, as this might be prohibitively time consuming. A rough estimate of the energy or enthalpy of the structure is required to filter out unfavorable structures, and a better energy ordering can be performed after the search is complete by thoroughly optimizing select systems of interest.
5. Once the optimization succeeds, create another set of input files with the desired convergence criteria for all degrees of freedom.
 6. Manually submit the new optimization step. If it fails, try the ideas above until it converges.
 7. Once the structure has converged to the desired level of accuracy, try to optimize another randomly generated structure using the optsteps that succeeded previously. Refine them if needed.
 8. Once you have successfully optimized enough random structures that you are confident in your method, gather all of the inputs used and write your scheme from them.

The scheme may be written by copying each input file into the template editor (with the appropriate optstep and template selected, of course) and replacing the structure-specific information with the appropriate keywords. Click the "Help" button for the complete list of keywords.

We have found that the optimization schemes are surprisingly transferable within an optimizer, so once you have a working optimization scheme for a given optimization code only minor tweaks (usually to the energy cutoffs, etc.) and appropriate adjustments to the volume limits and interatomic distances are necessary to use it on a different chemical system.

It is important to note that the optimization scheme does not have to perfectly converge your structures. A final post-processing optimization to refine any structure found in the search is highly recommended.

3.4 How to Save an Optimization Scheme for Later?

Once you have written your optimization scheme, you will want to save it for fast retrieval later (otherwise you will need to copy/paste and edit all of the templates again!). To save, simply click the "Save Opt Scheme" button and enter an appropriate filename with an extension of .scheme.

3.5 How to Load an Optimization Scheme?

Loading an optimization is quite simple – just click the "Load Opt Scheme" button and select the .scheme file you wish to load. This will also update the current queuing system and optimizer to those specified by the scheme.

3.6 What is Saved?

The optimization scheme files contain more than just the templates for each optstep. They also store queue and optimizer specific settings. This is useful for storing configuration options for different clusters along with the scheme. Note that although XtalOpt will prompt for an SSH password if needed, it is **NOT** stored in the scheme file.

3.7 Suggestions for Optimization Schemes

3.7.1 Crystals (XtalOpt)

The following list describes the optimization steps used in the `schemes/vasp-xtalopt.scheme` file distributed with the XtalOpt source code:

1. Fix unit cell, only optimize atomic coordinates. A very loose convergence criterion is used, and the number of KPOINTS is kept small.
2. The cell volume is fixed, but atomic positions and cell parameters are allowed to vary. The convergence criteria is the same as before, as is the KPOINT grid.
3. All degrees of freedom are considered using the same convergence criteria as before, but with a finer KPOINT grid.
4. Same as before, but with a stricter convergence criteria.
5. Same as before, but with a stricter convergence criteria and more KPOINTS.
6. Same as before, but with more KPOINTS.

This is only one of many possible optimization schemes that may work for crystals. It may need to be modified to work for your particular system.