XtalOpt User Guide

Version 13.2

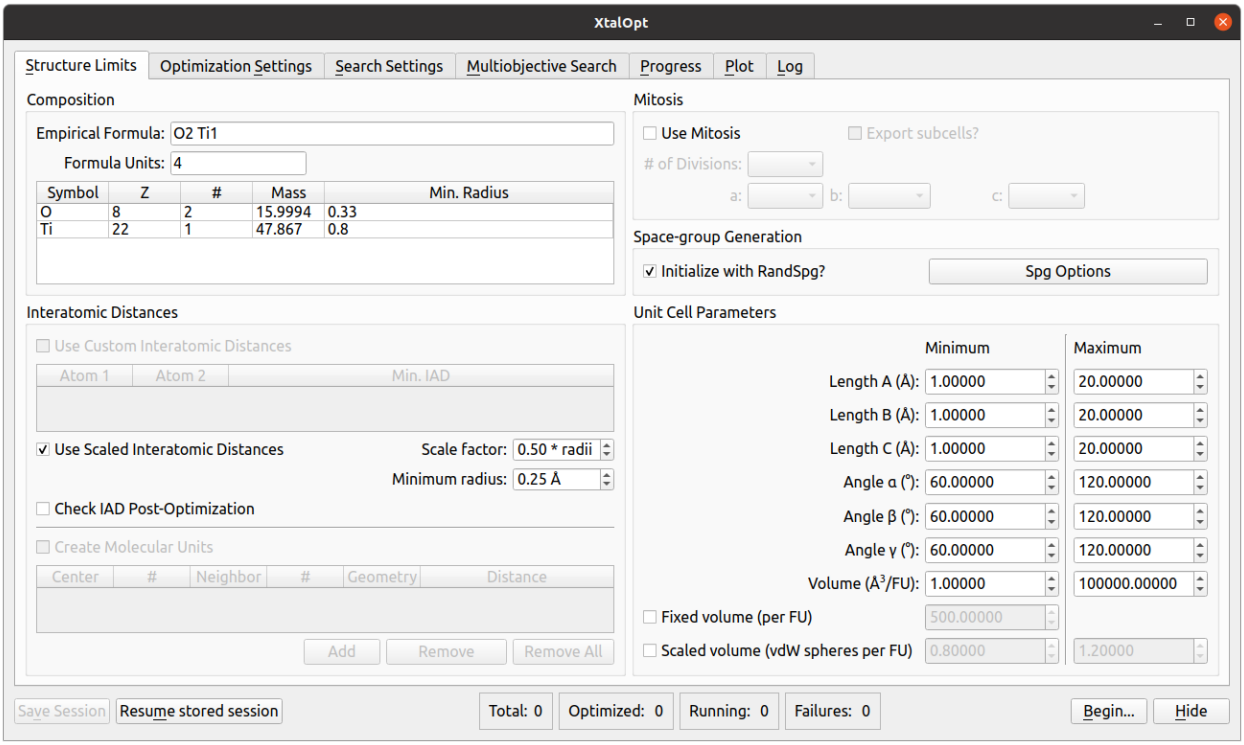
Generated by Doxygen 1.9.8

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

1.2.1 Chemical Composition

We will use a 6 formula unit supercell of titanium dioxide for this tutorial, so enter "Ti1 O2" for the "Empirical Formula". Under the "Formula Units" we can type 6. If the user wants to search multiple formula units, they would only have to type in the range, or individual numbers they wish to search (e.g., 1-6; or 1-2, 4, 6).

1.2.2 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., 0.8 and 1.2 for the minimum and maximum values, respectively).

Then, XtalOpt first calculates the total volume of spheres of van der Waals radius for all atoms in a formula unit. Then, it multiplies that total volume in 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.

In the command-line interface (CLI) mode of XtalOpt run, the user can invoke this functionality by specifying the pair of flags:

```
volumeScaleMin = ####
volumeScaleMax = ####
```

with the corresponding values. One can check the final calculated values in the run output, i.e., the "xtalopt.state" file (and "xtalopt-runtime-options.txt" file in the CLI mode).

1.2.3 Interatomic Distances (IAD)

There are now 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.

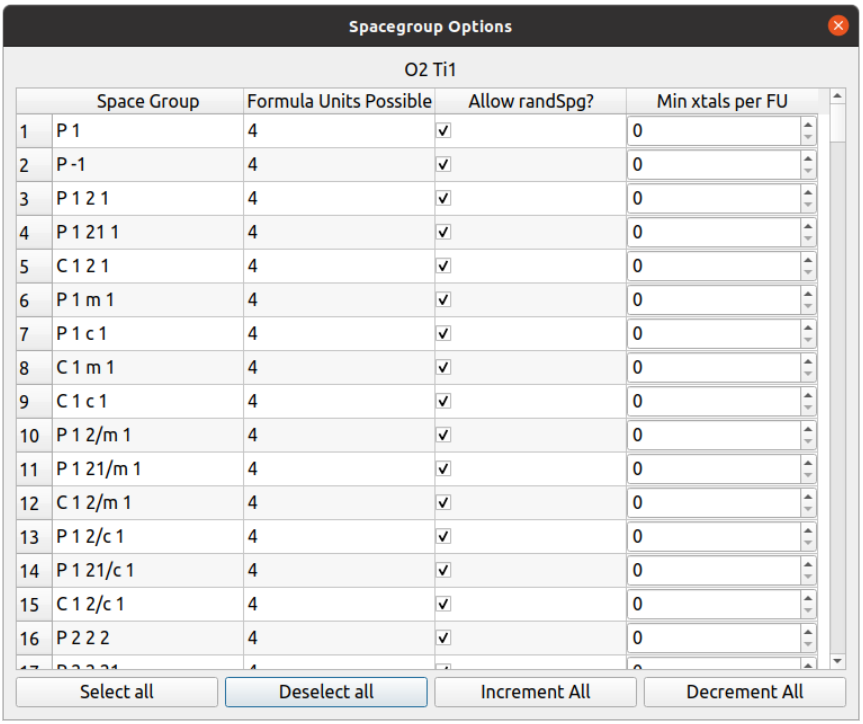
Finally, a checkbox labelled "Check IAD Post-Optimization" is also now available. If this box is checked, the interatomic distances are checked after the optimization is complete, and if any structures fail the interatomic distance check, they will be marked as failed structures.

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 composition has been defined, the user can define a single-center molecule that assumes one of the VSEPR geometries. For example, with the 6 formula unit TiO2, we could set 6 Ti atoms to be the center of 2 O atoms in a linear geometry. 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 untis are added.

The molecular units are only used in the initial generation.

1.2.5 Random Spacegroup Generator



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 chemical composition and number of formula units.

Spacegroup constraints can only be used in the initial generation. Randspg cannot be used simultaneously with molunit nor with mitosis.

For more information on RandSpg, see `this paper`.

1.2.6 Mitosis

For structures with large unit cells/large number of atoms, one can use "Mitosis" to increase the local order of the initially generated structures. This will create a small subcell, then multiply the subcell in each direction to fulfill the stoichiometry and unit cell size. "# of Divisions" defines how many subcells will be replicated, based upon the user defined number of "Formula Units", and the a, b, c determine in which direction the cells are replicated.

1.3 Optimizer Setup

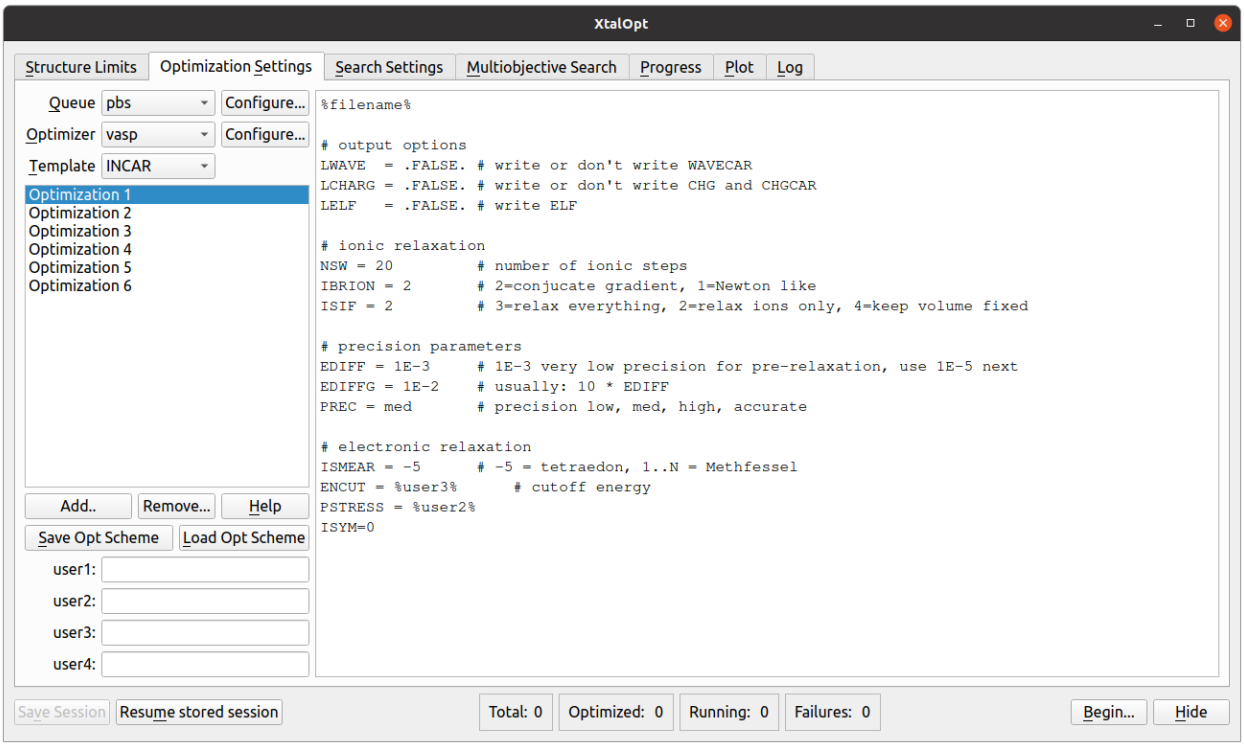
XtalOpt primarily supports the VASP, GULP, PWscf, CASTEP, and SIESTA codes for performing geometry optimizations. Each is detailed in its own section below.

As of release 12, however, XtalOpt now supports a generic optimizer, which can potentially be used for many different kinds of optimizers. The generic optimizer is unique in many ways, and its details 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 program installation is different, and it is almost certain that the submit files included with these schemes will not work on any cluster other than the Zurek group's "parity" 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.

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%
```

or, equivalently, in the CLI mode by adding a "system" type of POTCAR to the input file, i.e.,

```
potcarFile system = /path_to/potcar
```

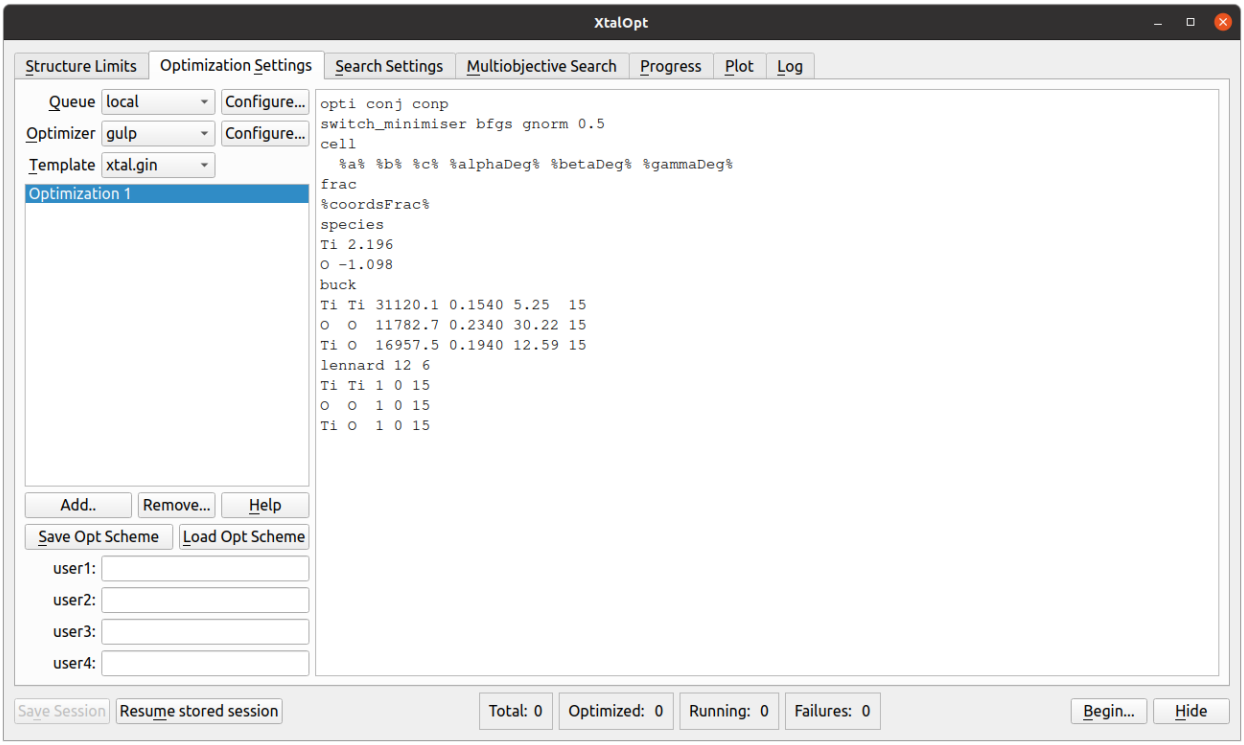
It should be noted that:

1. as XtalOpt arranges the chemical elements in alphabetical order, individual POTCAR files should be combined in the same order to prodcue 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.

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.

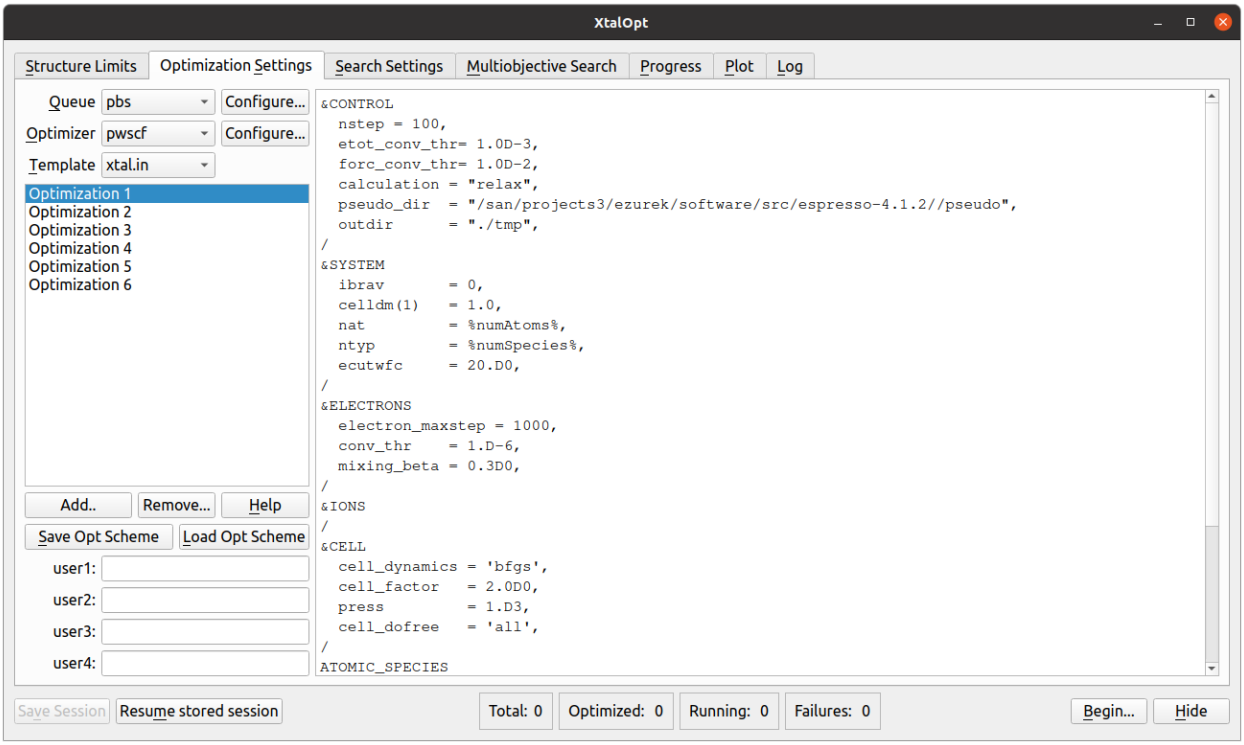
Note the "%" surrounding various keywords. These will be replaced by the structure-specific data when the optimizer is invoked for each structure. Click "Help" to view all of the keywords available. The number of optimization steps can be modified with the "Add/Resume" buttons. The "user" fields in the 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 "pwscf-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.

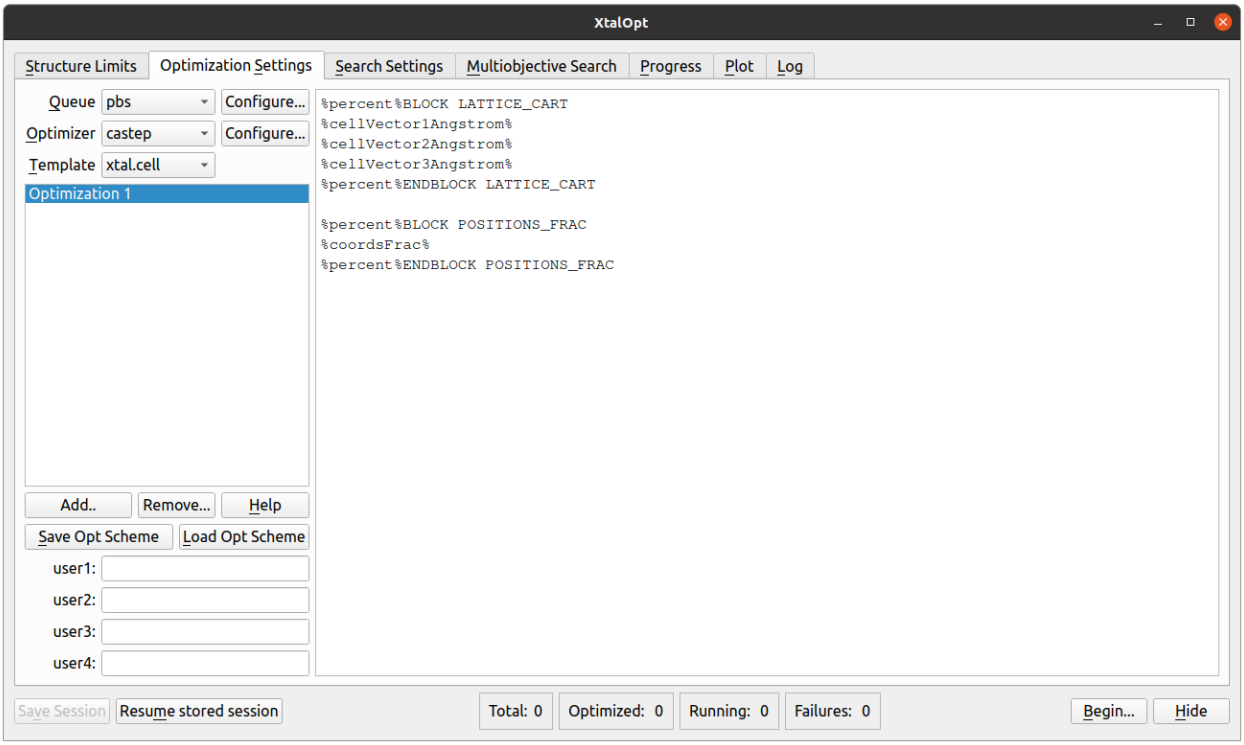
Notice the %keyword% values in the job.pbs templates. These are used to enter information that is specific to a search or structure when the actual input files are written prior to job submission. Click the "Help" button for a full listing of the available keywords.

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.

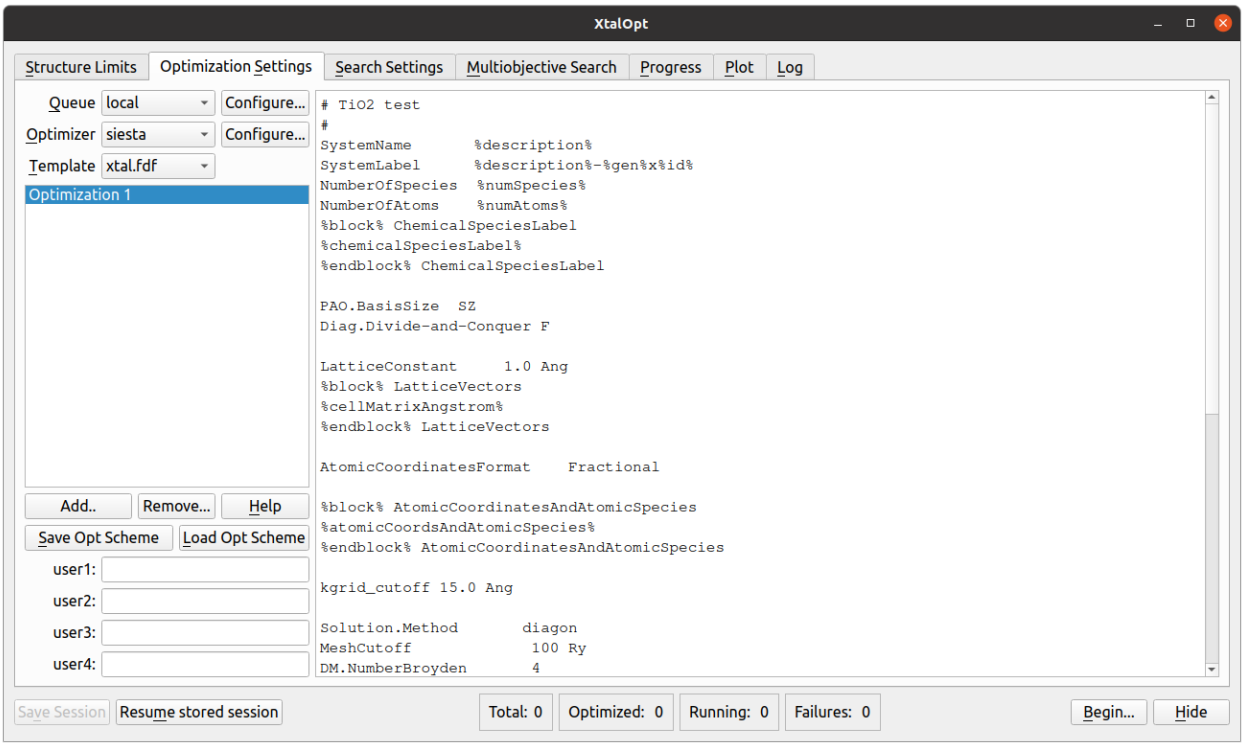
Notice the %keyword% values in the job.pbs templates. These are used to enter information that is specific to a search or structure when the actual input files are written prior to job submission. Click the "Help" button for a full listing of the available keywords.

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.

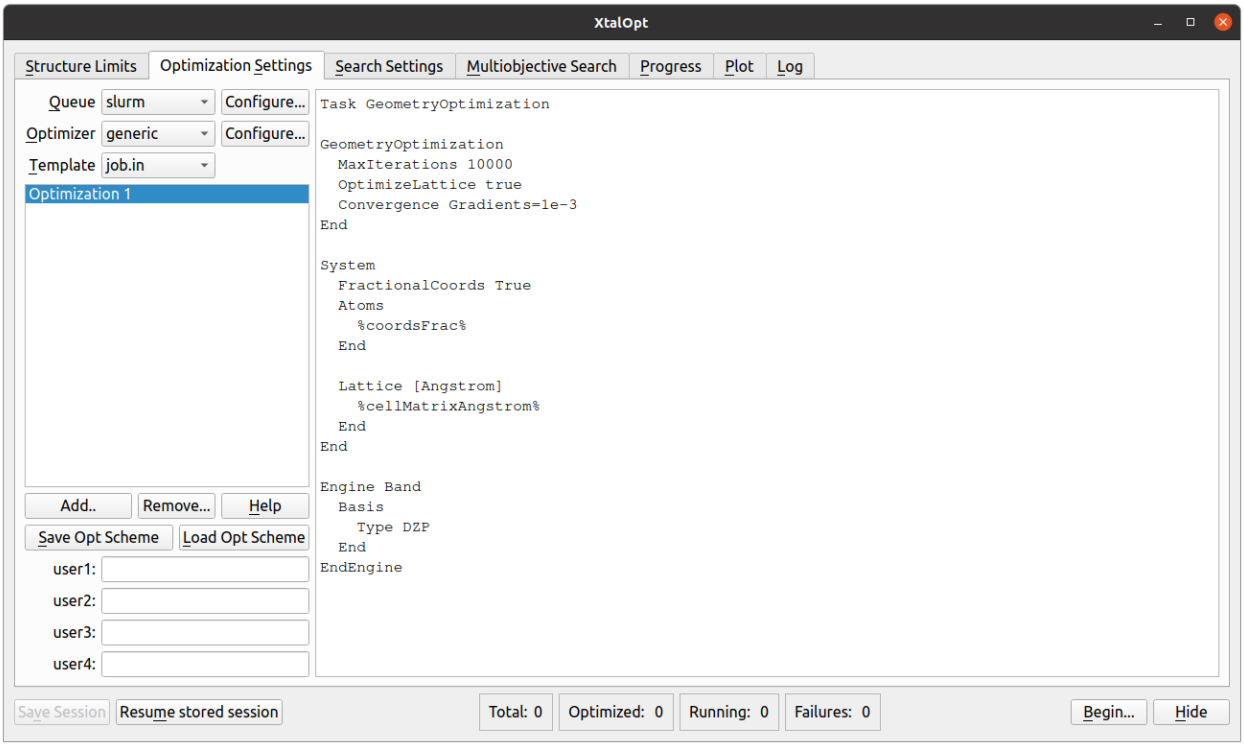
Notice the other %keyword% values in the xtal.fdf templates. These are used to enter information that is specific to a search or structure when the actual input files are written prior to job submission. Click the "Help" button for a full listing of the available keywords.

Note: that in the current implementation XtalOpt uses the "Total Final Energy" printed in the output to 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 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.

1.4 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.

New to 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.

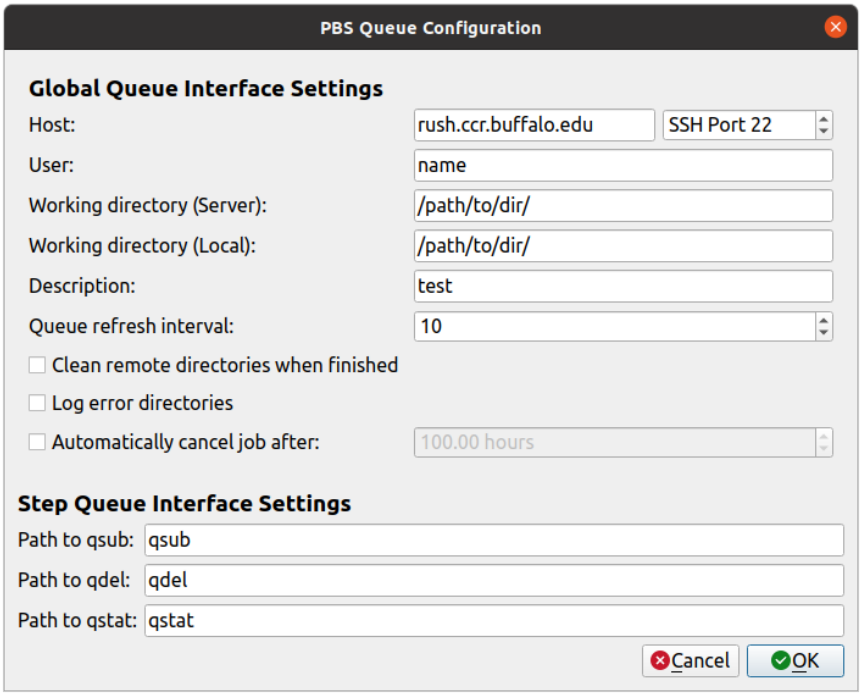
In addition, the queue configure menu (accessed via the "Configure..." button next to the queue interface) is now divided into two different parts: global queue interface settings and step queue interface settings. The global queue interface settings are settings for all optimization steps. The step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



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%x%id%-%optstep%.out
#PBS -e ../%gen%x%id%-%optstep%.err
#PBS -N %description%-%gen%x%id%-%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%
```

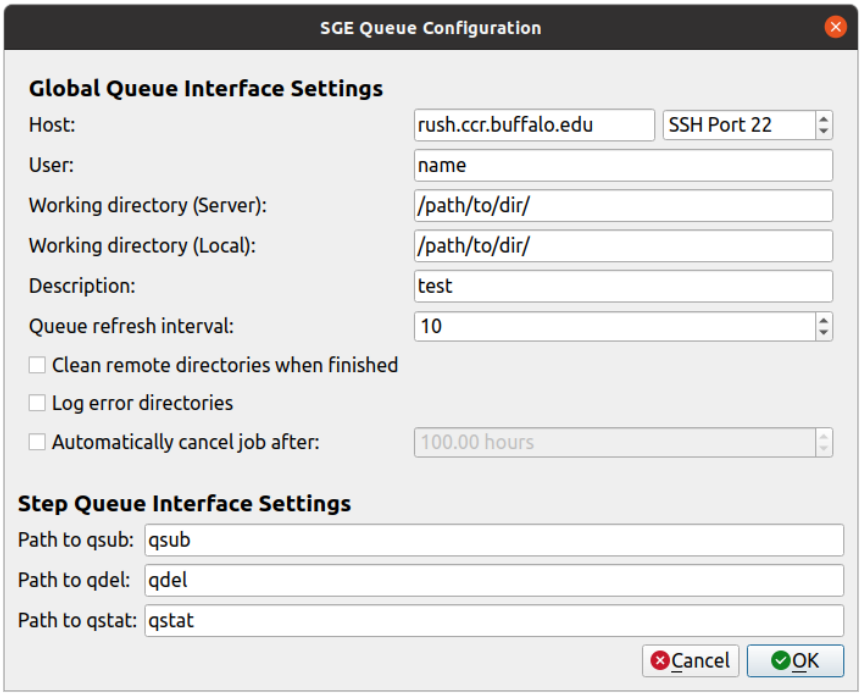
This will name each job, for example, xtalSearch-3x4-2, where xtalSearch is a user-specified description of the search, and 3x4-2 means that it is the fourth structure in the third generation running its second optimization step.

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



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

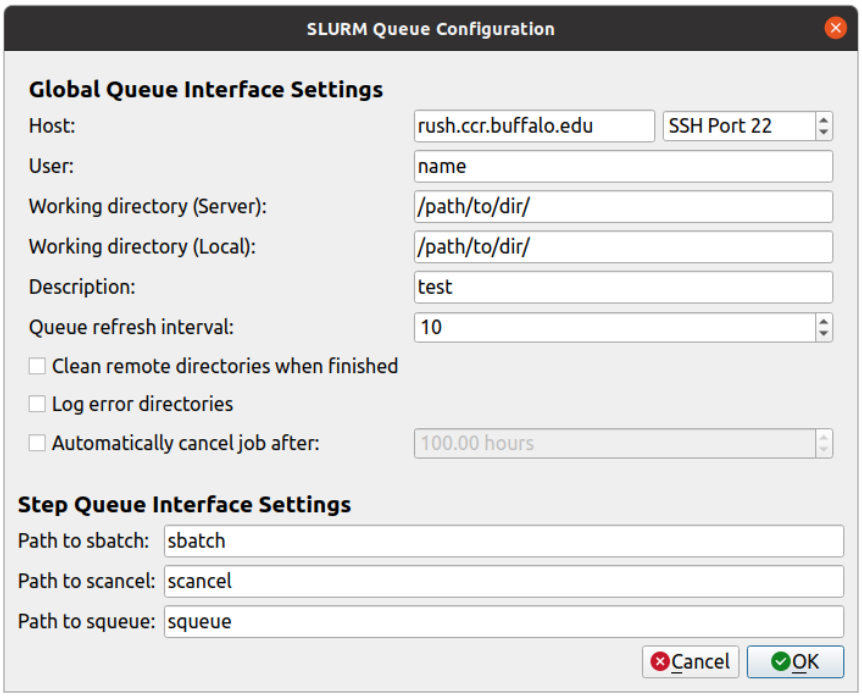
- Path to qsub: Where to find the qsub executable on the remote cluster. Note that if qsub is in the cluster's $PATH, setting this to just 'qsub' will work.
- Path to qdel: Where to find the qdel executable on the remote cluster. Note that if qdel is in the cluster's $PATH, setting this to just 'qdel'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



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.

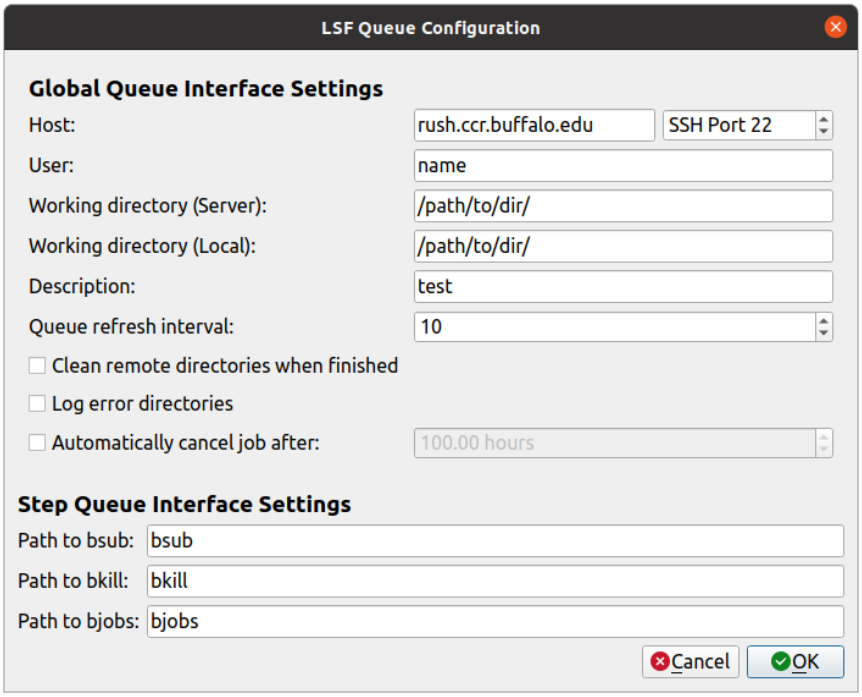
- Path to squeue: Where to find the squeue executable on the remote cluster. Note that if squeue is in the cluster's $PATH, setting this to just 'squeue' will work.

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



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

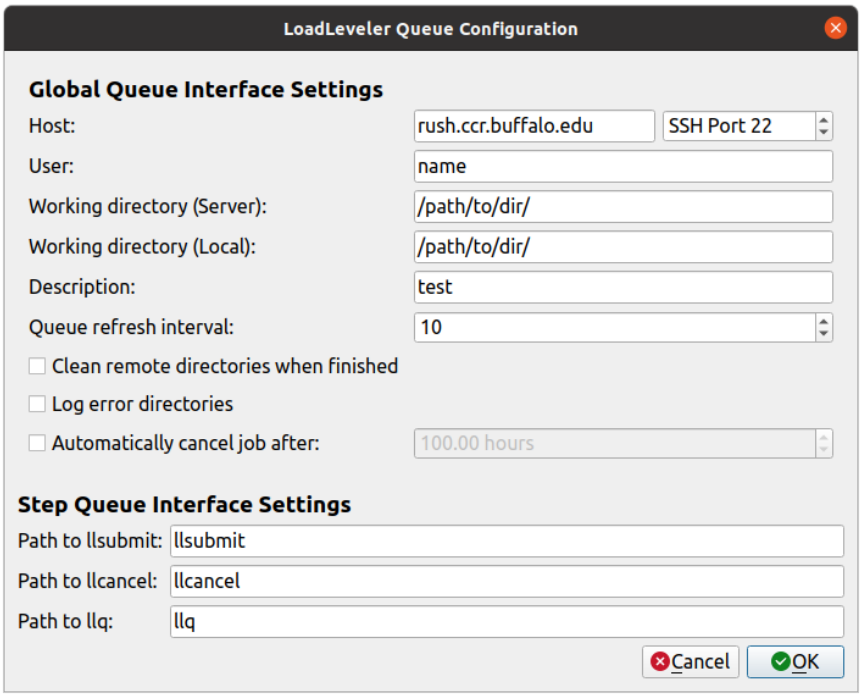
- Path to bsub: Where to find the bsub executable on the remote cluster. Note that if bsub is in the cluster's $PATH, setting this to just 'bsub' will work.
- Path to bkill: Where to find the bkill executable on the remote cluster. Note that if bkill is in the cluster's $PATH, setting this to just 'bkill'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



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

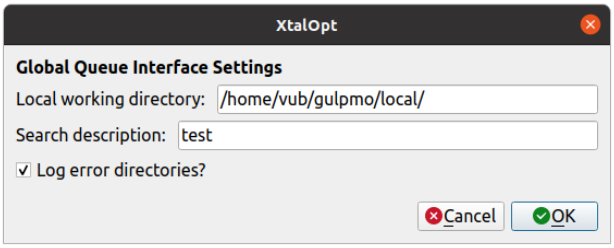
- Path to llsubmit: Where to find the llsubmit executable on the remote cluster. Note that if llsubmit is in the cluster's $PATH, setting this to just 'llsubmit' will work.
- Path to llcancel: Where to find the llcancel executable on the remote cluster. Note that if llcancel is in the cluster's $PATH,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

1.4.7 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.

1.5 What Is Written to the Local Directory?

A directory for each structure is created at

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

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

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

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

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

which 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_old.txt", 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.

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	Gen	ID	INDX	Enthalpy/FU	FU	SpaceGroup	Status
1	2	10	16	-39.8004	4	P4_2/mnm	Optimized
2	2	7	13	-39.8004	2	P4_2/mnm	Skipped Optimization
3	1	7	6	-39.8004	4	P4_2/mnm	Supercell (2x7)
4	2	2	8	-39.8004	4	P4_2/mnm	Supercell (2x7)
5	2	9	15	-39.8004	4	P4_2/mnm	Optimized
6	2	4	10	-39.8004	4	P4_2/mnm	Supercell (2x7)
7	3	1	18	-39.8004	2	P4_2/mnm	Duplicate (2x7)
8	1	3	2	-39.5173	4	P-1	Optimized
9	2	8	14	-39.2212	4	C2/m	Supercell (2x1)
10	2	1	7	-39.2212	1	P-1	Skipped Optimization
11	1	2	1	-39.2212	4	P-1	Supercell (2x1)
12	2	3	9	-39.1877	4	P-1	Optimized
13	2	12	19	-39.1809	4	P2_1	Optimized
14	2	6	12	-38.9898	4	P1	Optimized
15	1	4	3	-38.9372	4	P2/m	Optimized
16	1	1	0	-38.8513	4	P1	Optimized
17	2	5	11	-38.764	2	P-1	Skipped Optimization
18	1	5	4	-38.764	4	P-1	Supercell (2x5)
19	2	11	17	-38.4902	4	P1	Optimized
20	1	6	5	-38.0696	4	P1	Optimized

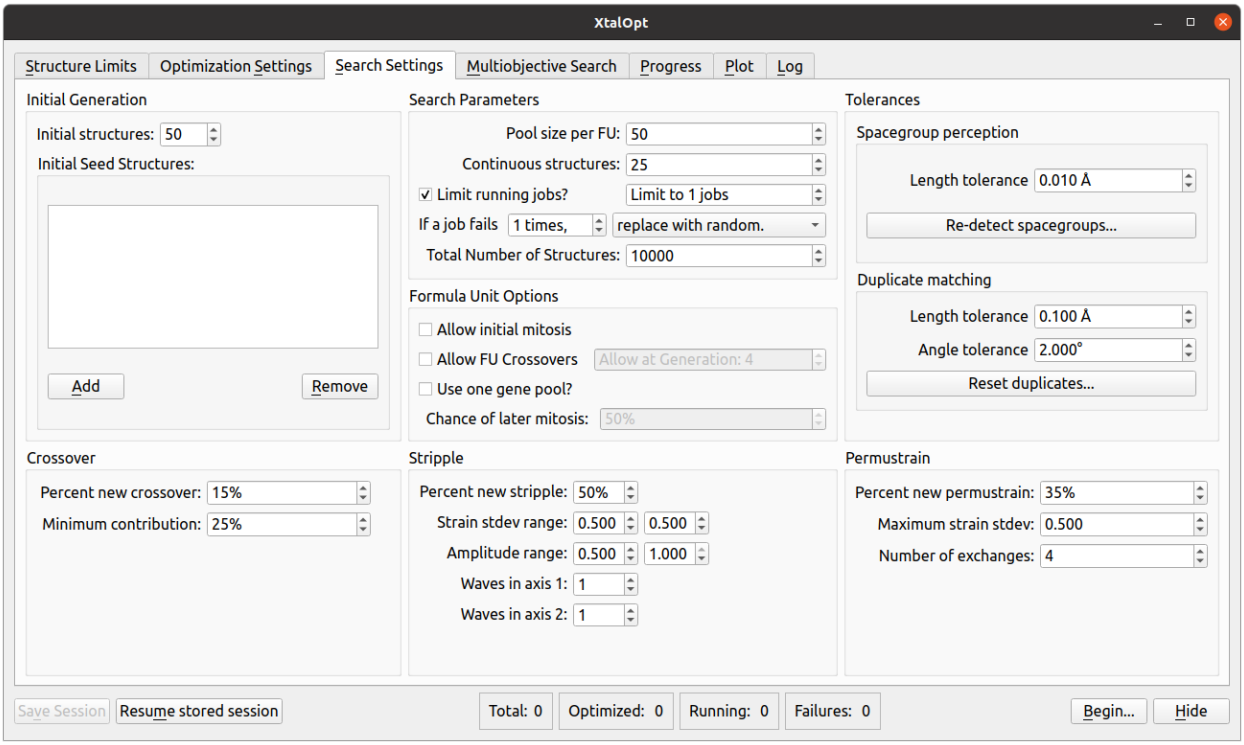
The following is a description of each column:

- Rank: The rank from the lowest enthalpy per formula unit (low numbers) to the highest enthalpy per formula unit (high numbers).

- Gen: The generation number of the structure.
- ID: The ID (or offspring number) of the structure.
- INDX: The order in which structures are generated (0 is the first structure, etc).
- Enthalpy/FU: The enthalpy per formula unit of the structure.
- FU: The formula unit of the structure.
- SpaceGroup: The Hermann-Mauguin symbol of the space group for the structure.
- Status: The current status of the structure. If the status is a duplicate/supercell, the "Tag" of the structure of which it is a duplicate/supercell will follow (i.e., in parentheses as <generation>x<idNumber>). If the status says "Skipped Optimization", then, most likely, the structure is a primitive reduction of another structure.

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).

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 5, 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 running jobs unless running XtalOpt locally. Once running the code on a cluster, the queueing system on the cluster 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 new structure. In the CLI mode, however, it is possible to instruct XtalOpt to quit after generating the specified number of structures (see Terminating XtalOpt Runs for details).

The tolerances for duplicate matching 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.

1.7 Multiobjective Search

The multiobjective search in the XtalOpt code is designed to simultaneously optimize 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 (calculated using AFLOW-ML), symmetry (space group number), volume per atom, synthesizability likelihood, superconducting critical temperature, or any property that can be represented by a single numerical value. XtalOpt calculates a generalized fitness function using these values, according to the optimization type of each objective (minimization, maximization, etc.) as specified by the user.

This functionality is available in both the CLI and GUI modes of the XtalOpt code. In general, the user can define an arbitrary number of objectives for a multiobjective search and run the search on either a local computer or a remote queue.

1.7.1 Multiobjective Search Parameters

For any property that the user wants to use as a objective in the multiobjective 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 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 multiobjective 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 (samples are provided in this document).

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 multiobjective functionality, for every desired objective, the user should provide the following information to the XtalOpt code:

- **optimization 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, maximize the AFLOW-ML hardness, 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".

1.7.2 Examples of user-defined scripts

The following is an example of a script that can be employed with both the CLI and GUI versions of XtalOpt. 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. In the XtalOpt input file for the CLI mode, this can be then introduced as:

```
objective = max /path/spg.sh spg.dat
```

along with the desired weight (or leaving the weight un- specified for XtalOpt to adjust it).

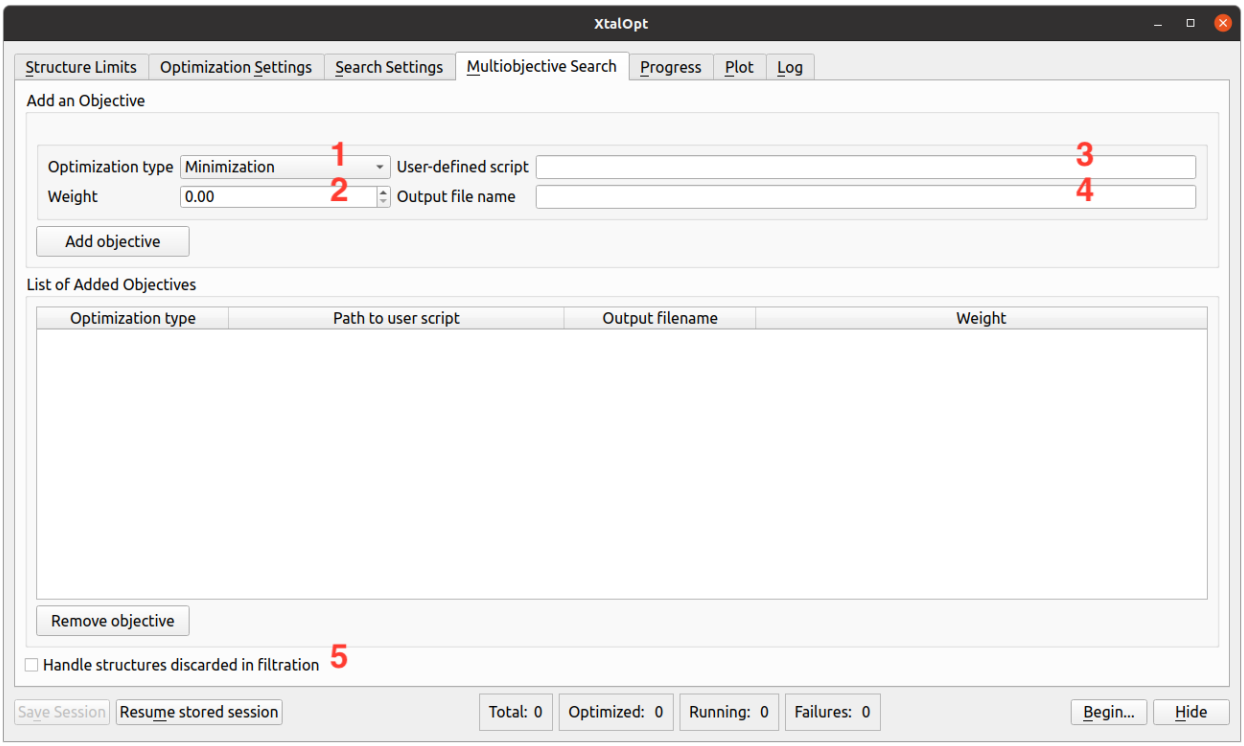
Alternatively, if the user desires to submit this calculation to a computational cluster, the executable script /path/spg queue.sh in its most basic form can be written as:

```
#!/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 /paht/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 multiobjective 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.

1.7.3 Multi-objective Runs in the XtalOpt GUI

In the version 13.0 of XtalOpt a new Multiobjective Search tab is introduced, where the user can add desired objectives to be optimized along with the enthalpy.



In the Multiobjective Search tab, all entries relevant to a multiobjective run (described above) can be entered in the corresponding fields by:

1. choosing the optimization type for the objective from a drop-down menu, i.e., "Minimization", "Maximization", "Filtration" (see Filtering Structures: Constrained Search), and "Hardness" (see Hardness Optimization),
2. entering or setting the weight, and
3. entering the full path to the external code (or script) that calcualtes the objective,
4. entering the output file name that the external code generates with the objective value,
5. specifying whether a structure discarded by a filtration feature requires further handling or not (see Filtering Structures: Constrained Search for more details).

Then, the selected objective can be added to the list of objectives for the run.

Initializing the multiobjective search in the CLI mode is discussed in the Multiobjective Runs in the XtalOpt CLI section.

1.7.4 Filtering Structures: Constrained Search

The multiobjective 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.

To utilize the filtration functionality, similar to the "minimization" and "maximization" features, 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 (or, in the CLI mode, the value of the jobFailAction input flag). 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.

For instructions of utilizing the constrained search in the CLI mode, see the Multiobjective Runs in the XtalOpt CLI section.

1.7.5 Hardness Optimization

Since XtalOpt 12, maximizing the AFLOW-ML hardness could be invoked with initiating relevant entries in the Search Settings tab (or, by using the calculateHardness flag in the CLI version of the XtalOpt code). After obtaining the predicted shear modulus (G) from the AFLOW-ML for a structure, XtalOpt would calculate the Vickers hardness (HV) from the Teter model, i.e.,

$$HV = 0.151 * G$$

As of XtalOpt version 13.0, the handling of the maximizing the AFLOW-ML hardness has changed:in the GUI mode, the "Search Settings" tab does not include the hardness entries anymore. Instead, the user can choose an objective of the "hardness" type in the "Multiobjective Search" tab and assign the desired weigh for that objective (for the CLI mode settings, see the Multiobjective Runs in the XtalOpt CLI section).

Therefore, AFLOW-ML hardness calculations are now treated as a user-defined objective. It should be noted, however, that for a "hardness" objecive:

- the script name and the output file name inputs do not need to be provided,
- while an arbitrary number of objectives with "maximization", "minimization", and "filtration" type can be introduced, no more than one "hardness" objective can be added by the user.

It should be noted that an objective of the "hardness" type performs the hardness optimization by obtaining the relevant data from AFLOW-ML through internal functions of the XtalOpt code. This is a legacy code and will be disabled in future releases of XtalOpt to avoid compatibility issues. Instead of using this type of objective, users are strongly encouraged to use a script to facilitate AFLOW-ML hardness optimization, similar to any regular "maximization" 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 the AFLOW-ML (and, e.g., further process 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
```

1.7.6 XtalOpt Output in Multiobjective Runs

Generally, during a XtalOpt run, the search settings are written to the "xtalopt.state" (and "xtaloptSettings.log" in the CLI mode) file. In a multiobjective run, these files will include the multiobjective search settings.

As described in Reading the results.txt File, the "results.txt" file shows a live status of all structures generated during the evolutionary search run.

In a multiobjective 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 "ObjectiveCalculation". Once the objectives' values are calculated, the status changes to "Optimized", "ObjectiveDismiss", or "ObjectiveFail" 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 the legacy "hardness" objective, the corresponding column in the "results.txt" file has the title of "Hardness". If the hardness of the structure has not yet been calculated, "-1" will be shown. If the user did not choose to calculate the hardness in the options, the hardness column will be omitted from the results.txt file.

Finally, 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.

1.8 "Begin"

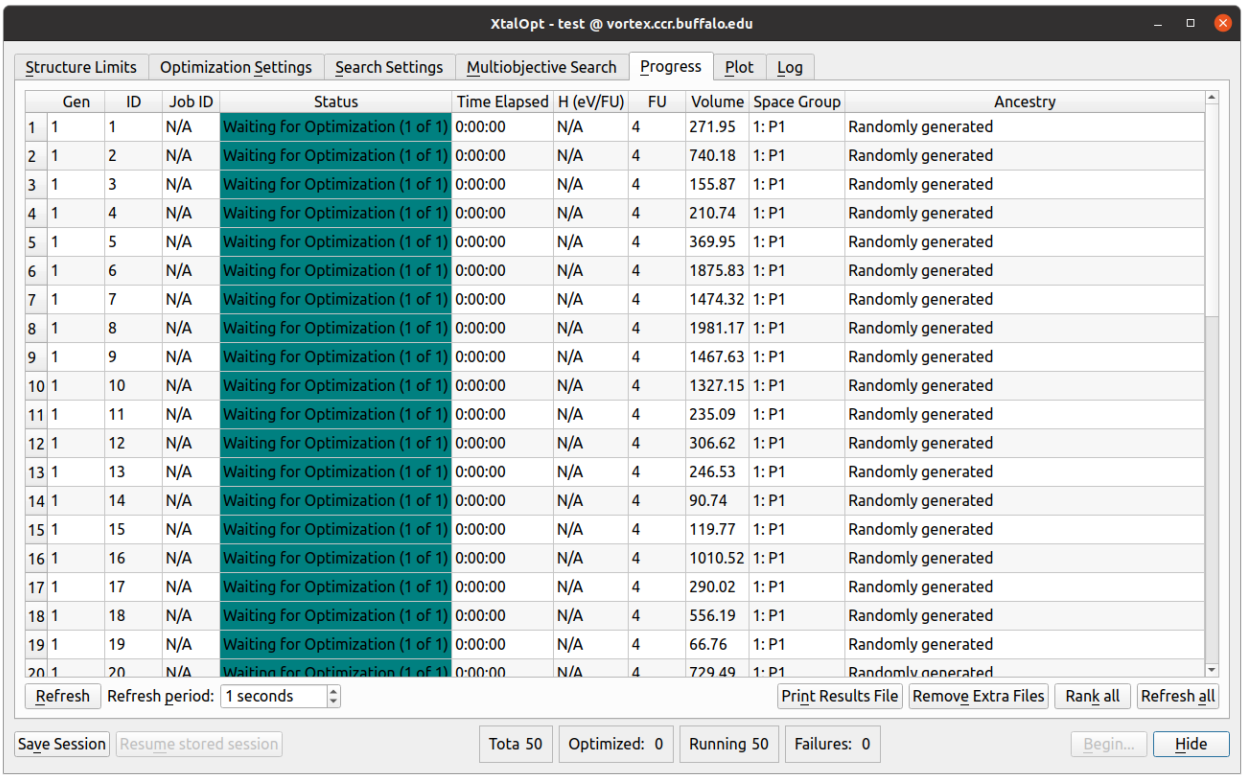
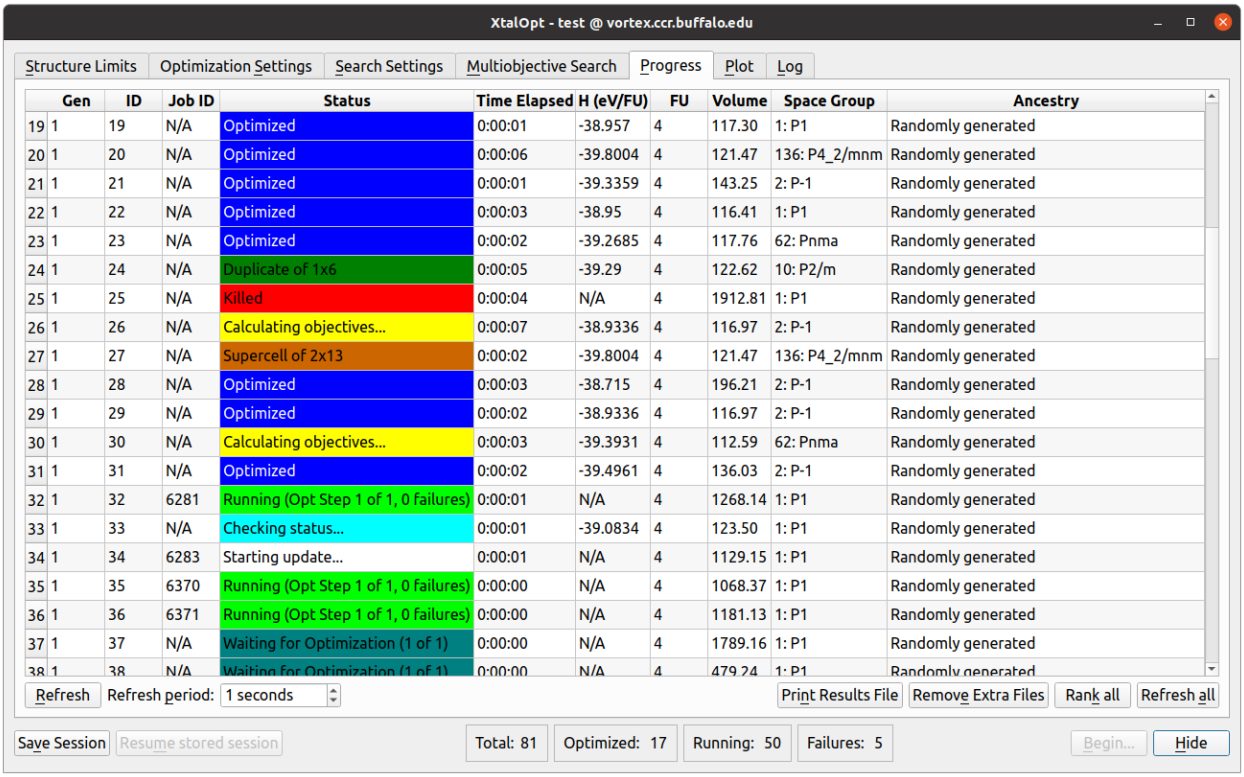


Figure 1 The ``Progress'' tab immediately after starting a search

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



	Gen	ID	Job ID	Status	Time Elapsed	H (eV/FU)	FU	Volume	Space Group	Ancestry
19	1	19	N/A	Optimized	0:00:01	-38.957	4	117.30	1: P1	Randomly generated
20	1	20	N/A	Optimized	0:00:06	-39.8004	4	121.47	136: P4_2/mnm	Randomly generated
21	1	21	N/A	Optimized	0:00:01	-39.3359	4	143.25	2: P-1	Randomly generated
22	1	22	N/A	Optimized	0:00:03	-38.95	4	116.41	1: P1	Randomly generated
23	1	23	N/A	Optimized	0:00:02	-39.2685	4	117.76	62: Pnma	Randomly generated
24	1	24	N/A	Duplicate of 1x6	0:00:05	-39.29	4	122.62	10: P2/m	Randomly generated
25	1	25	N/A	Killed	0:00:04	N/A	4	1912.81	1: P1	Randomly generated
26	1	26	N/A	Calculating objectives...	0:00:07	-38.9336	4	116.97	2: P-1	Randomly generated
27	1	27	N/A	Supercell of 2x13	0:00:02	-39.8004	4	121.47	136: P4_2/mnm	Randomly generated
28	1	28	N/A	Optimized	0:00:03	-38.715	4	196.21	2: P-1	Randomly generated
29	1	29	N/A	Optimized	0:00:02	-38.9336	4	116.97	2: P-1	Randomly generated
30	1	30	N/A	Calculating objectives...	0:00:03	-39.3931	4	112.59	62: Pnma	Randomly generated
31	1	31	N/A	Optimized	0:00:02	-39.4961	4	136.03	2: P-1	Randomly generated
32	1	32	6281	Running (Opt Step 1 of 1, 0 failures)	0:00:01	N/A	4	1268.14	1: P1	Randomly generated
33	1	33	N/A	Checking status...	0:00:01	-39.0834	4	123.50	1: P1	Randomly generated
34	1	34	6283	Starting update...	0:00:01	N/A	4	1129.15	1: P1	Randomly generated
35	1	35	6370	Running (Opt Step 1 of 1, 0 failures)	0:00:00	N/A	4	1068.37	1: P1	Randomly generated
36	1	36	6371	Running (Opt Step 1 of 1, 0 failures)	0:00:00	N/A	4	1181.13	1: P1	Randomly generated
37	1	37	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	4	1789.16	1: P1	Randomly generated
38	1	38	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	4	479.24	1: P1	Randomly generated

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 2x2 has been automatically marked as a duplicate (dark green) of structure 1x2 and removed from the breeding pool, structure 3x4 is currently undergoing a local optimization (light green), while structure 2x10 is waiting to be optimized (light blue).

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.

After every structure has been optimized, it is checked to see if it is a super cell by performing a primitive reduction. If the primitive reduction yields a crystal with fewer atoms than it originally had, the original crystal is marked as a super cell and a new crystal is generated that is labelled as a primitive reduction of the original. This primitive-reduced crystal does not undergo optimization - it is labelled with the same energy as the original crystal. And, if the user is searching the formula units of the primitive-reduced crystal, the primitive-reduced crystal becomes a part of that smaller formula unit's gene pool.

In addition, every time a new crystal is optimized, if the user is searching multiple formula units and the crystal is found to be low in energy, a supercell of the crystal may be generated and be added to the higher formula unit's gene pool.

In the case of a multiobjective 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 Avogadro2 section) – 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 "Print Results File" button generates a run-results.txt file that lists all of the information about each structure in order of generation and structure number (As compared to the results.txt file which ranks the structures). 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).

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.

1.9.1 View Trends

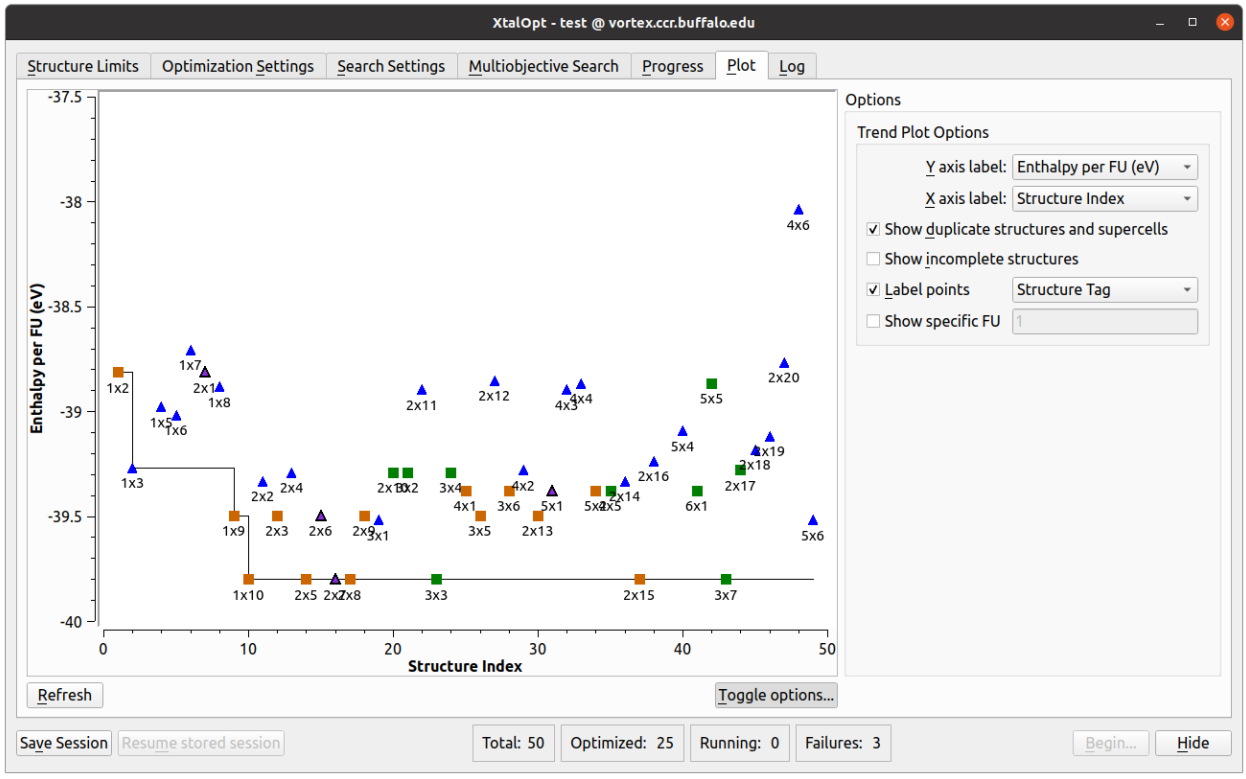


Figure 3 The ``Plot'' tab mid-run displaying enthalpy vs. structure index. Each structure is labeled with its Structure ID.

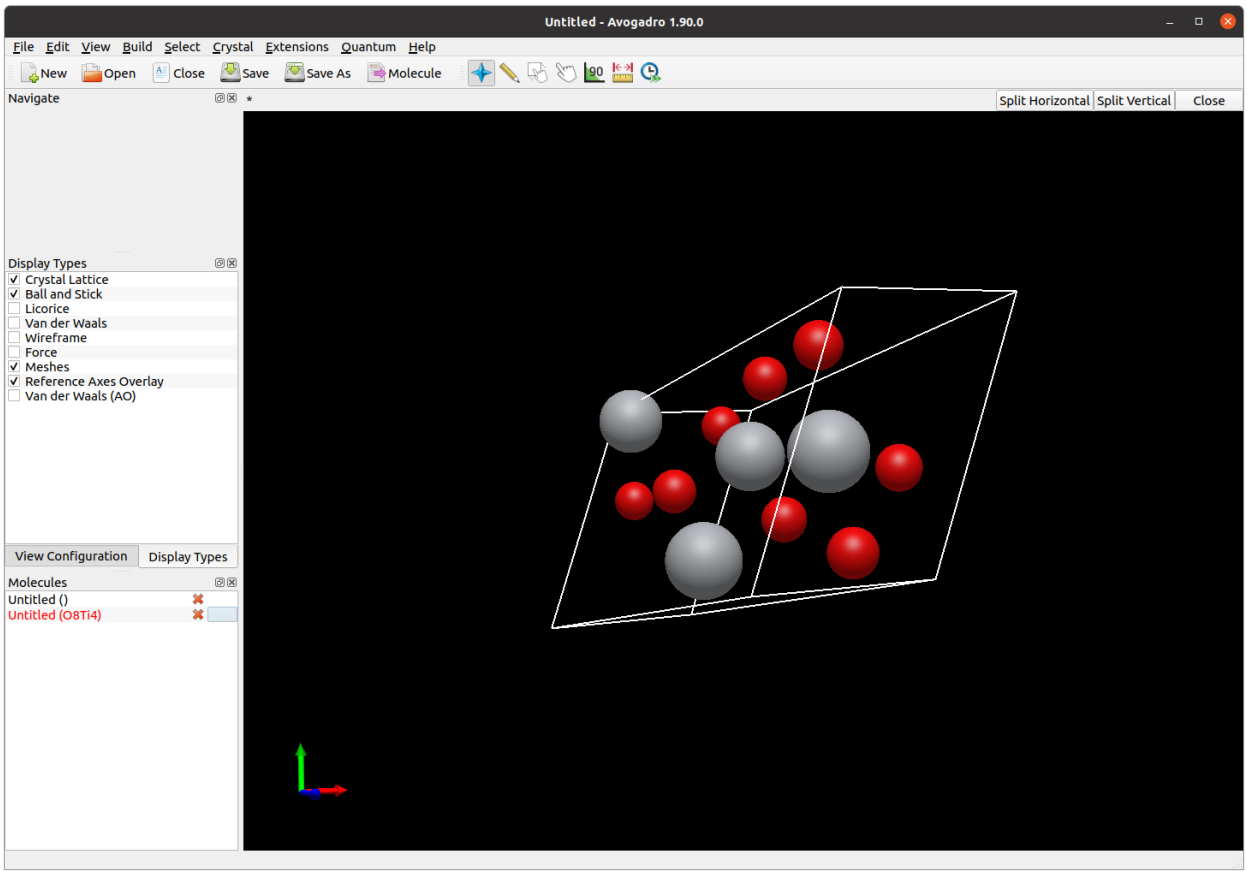
Another visualization and analysis tool available during the search is the interactive plot. The plot is capable of investigating trends in the search by plotting a point for each individual using structure number, generation number, enthalpy,

energy, PV enthalpy term, lattice parameters, or cell volume 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.

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 Avogadro2 section), allowing all the same functionality as described above in Monitor Progress.

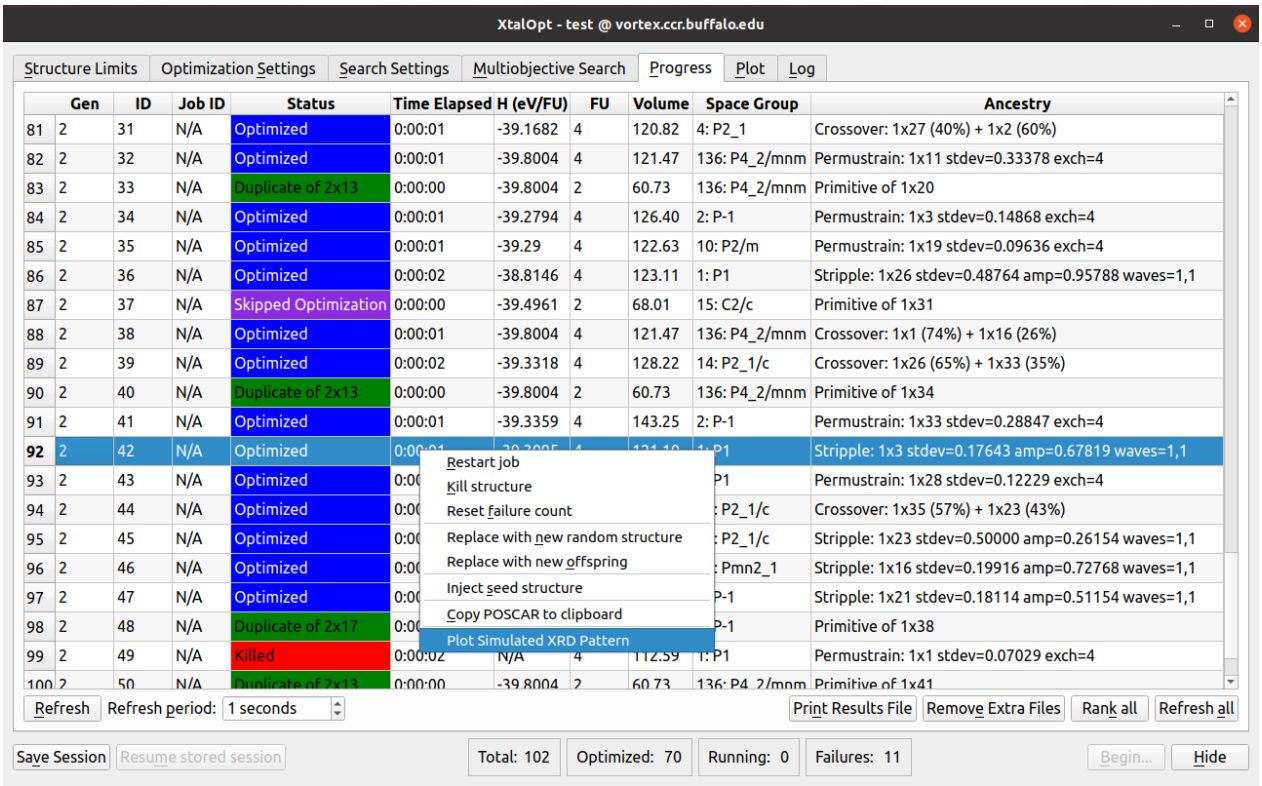
1.9.2 View Crystals in Avogadro2



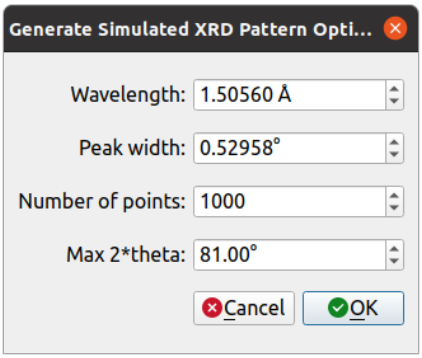
As of release 11, while an XtalOpt window is open, the user may easily view the crystals in Avogadro2. As long as an Avogadro2 window is open and an Avogadro2 Remote Procedure Call (RPC) server is running, when a user selects crystals in XtalOpt (either via the plot tab or the progress tab), the structure 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

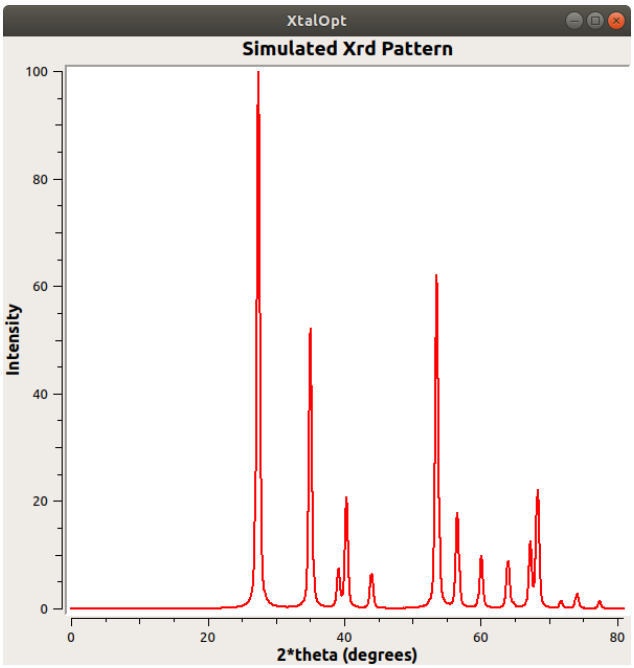
As of release 12, a simulated XRD pattern can be generated for any crystal using `Objcryst`. A user must simply right click on any entry in the progress tab, and then click "Plot Simulated XRD Pattern" (as shown in the image below).



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.

First of all, one can see all of the XtalOpt CLI options by running "xtalopt --help" from a terminal. This prints all options and a description of each one.

When performing a CLI run, an input file for XtalOpt is required. An example that contains detailed explanations of various options 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. If a different input file is desired, the user may explicitly select an input file with "--input-file [file]". The vast majority of the options are taken from the GUI, and more information on these options can be found in their respective places in the manual. One difference is that template files 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.

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 "xtaloptSettings.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 "xtalopt-runtime-options.txt". The settings in the "xtalopt-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.

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 the enthalpy/energy 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.

As of XtalOpt version 13, the user can add seed structures in the CLI mode as well. This can be done by introducing a "space separated" list of full path to the seed structures through the following flag in the XtalOpt input file:

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

1.10.1 Multiobjective Runs in the XtalOpt CLI

In XtalOpt version 13.0, the multiobjective functionality is implemented, as discussed previously in Multiobjective Search section. In order to invoke the multiobjective functionality in the CLI mode, 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 the objective and, generally, has the following format:

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

It should be noted that in the CLI mode of XtalOpt:

- Possible options for the "optimization type" are "minimization", "maximization", "hardness", and "filtration", as introduced above. This field is not casesensitive, and only the first three letters are important in identifying the optimization type by XtalOpt (i.e., "min", "max", "har", and "fil").
- Providing the "script output filename" is optional. If this is not specified, the default will be "objective#.out" in which "#" is the number of the objective in the order that it appears in the XtalOpt input file (excluding the "hardness" objective), e.g., "objective1.out", "objective2.out", etc.
- Specifying the "weight" for the objective is optional, as well. If this field is not given for a number of objectives, it will be determined via the following formula. If any weight is provided for any of the objectives, it will be subtracted from 1.0 and the remaining value will be divided between the enthalpy and objectives that don't have a specified weight.

To utilize the filtration functionality, the relevant entry in the XtalOpt input file for CLI mode can be:

```
objective = fil /path_to/script output_filename
```

In the CLI mode, the user can instruct XtalOpt to handle a structure which 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.

Further, AFLOW-ML hardness calculations can be invoked in the CLI mode using the input flag:

```
objective = hardness "weight"
```

Finally, in any XtalOpt run in the CLI mode the "xtalopt-runtime-options.txt" file which includes the parameters that can be modified during the search, is produced. Among the multiobjective-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).

1.11 Terminating XtalOpt Runs

In a regular XtalOpt run, and once the maximum number of structures (specified by the user) is generated, the user can resume the run by increasing this maximum number (among other run-time flags that can be changed). This functionality requires the code to not quit automatically, and the user needs to terminate the application manually after the desired output is obtained. There are, however, situations that the user prefers the code to exit after producing a specified number of structures (e.g., in a systematic run for multiple input files).

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
```

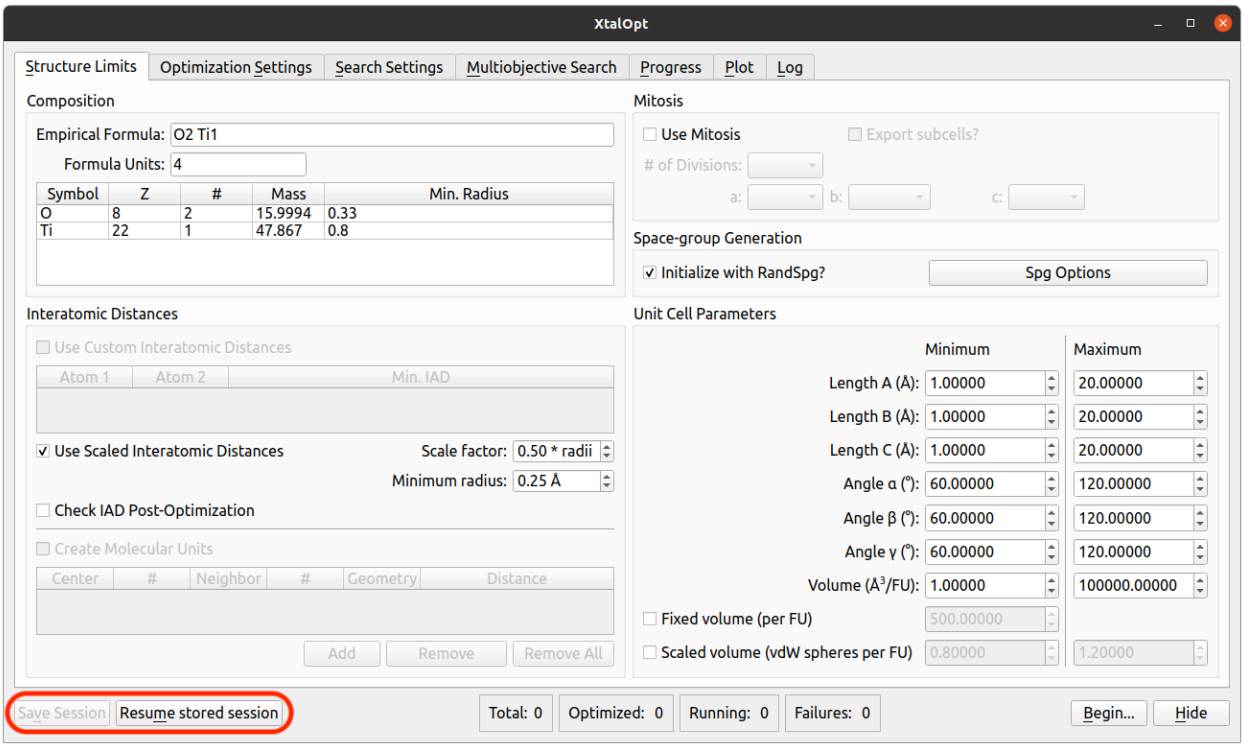
or by setting its value to true in the run-time setting file xtalopt-runtime-options.txt during the run. With this flag set to true, the code quits after all running (and pending) jobs are finished and the output files are updated.

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 xtalopt-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.

2 Saving and Resuming Sessions in XtalOpt



2.1 How to save your session

XtalOpt will write a small file named xtalopt.state to its working directory that contains the information necessary to resume the session at a later time. The file can be rewritten manually by clicking the "Save Session" button highlighted above, and XtalOpt will automatically save the session every time a structure is updated.

XtalOpt will also write a file "structure.state" in each candidate structure's directory. This file stores XtalOpt-specific information about the structure.

2.2 How to resume your session

To resume a session, simply click "Resume stored session" (highlighted above) and select the xtalopt.state file in the working directory of the session you would like to resume. XtalOpt will then begin to load the structures and search parameters. You can monitor the progress with the progress bar that appears at the bottom of the window.

While the structures are loading, you may encounter errors that say:

```
Error, no (or not appropriate for [OPTIMIZER]) xtal data in [DIRECTORY].

This could be a result of resuming a structure that has not yet done
any local optimizations. If so, safely ignore this message.
```

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 xtalopt.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 SM, Catlow CRA. Comp. Mat. Sci. 2009;45(1):84-95 (Available at: http://linkinghub.elsevier.com/retrieve/pii/S0927025608003030).

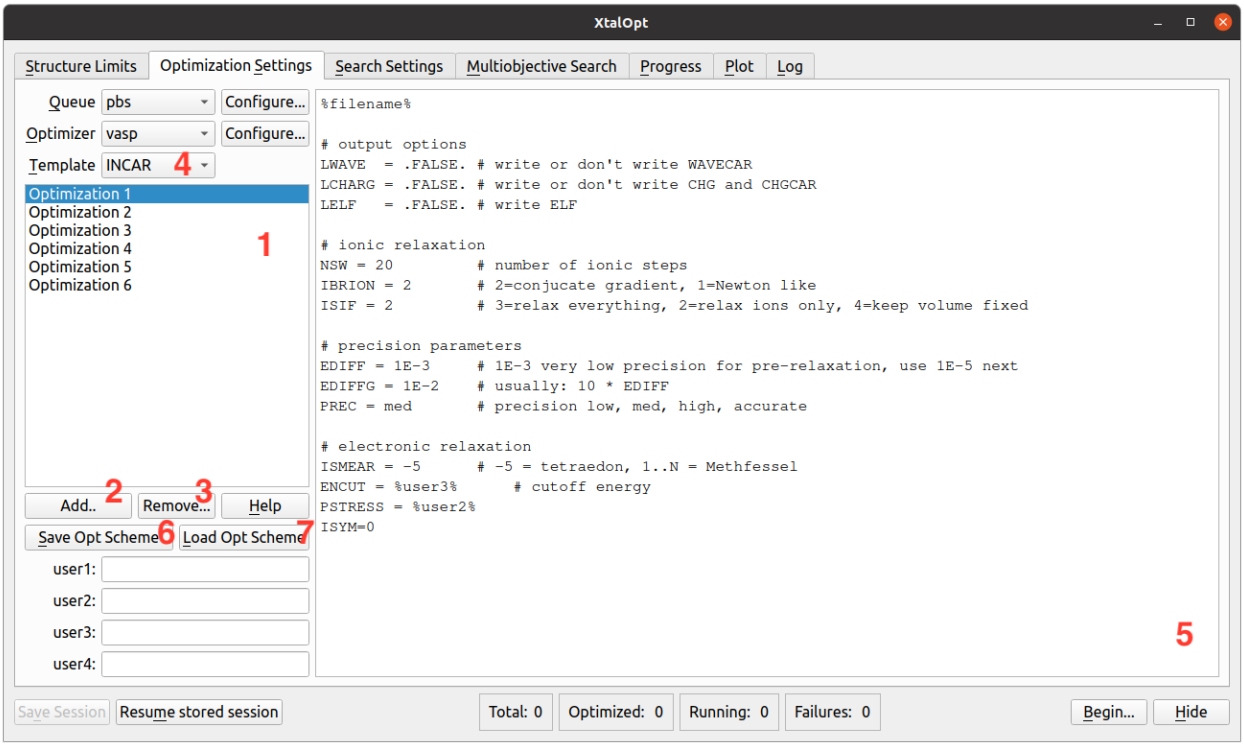
Why not just perform a single geometry optimization on each structure? Stochastic search techniques, such as XtalOpt, will often need to perform geometry optimizations on structures that are far from a stationary point on the potential energy surface. For example, the randomly generated structures in 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.

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 courser integration grid, etc).

(e) Change the minimizer (e.g. tell the optimizer to use conjugate gradients rather than BFGS, etc).

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.) 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.

Index