XtalOpt: An Open–Source Evolutionary Algorithm for Crystal Structure Prediction

David C. Lonie[a], Eva Zurek[a,*]

[a]*Department of Chemistry, State University of New York at Buffalo, Buffalo, New York, 14260-3000*

Abstract

The implementation and testing of XtalOpt, an evolutionary algorithm for crystal structure prediction, is outlined. We present our new periodic displacement (ripple) operator which is ideally suited to extended systems. It is demonstrated that hybrid operators, which combine two pure operators, reduce the number of duplicate structures in the search. This allows for better exploration of the potential energy surface of the system in question, while simultaneously zooming in on the most promising regions. A continuous workflow, which makes better use of computational resources as compared to traditional generation based algorithms, is employed. Various parameters in XtalOpt are optimized using a novel benchmarking scheme. XtalOpt is available under the GNU Public License, has been interfaced with various codes commonly used to study extended systems, and has an easy to use, intuitive graphical interface.

Keywords: Structure Prediction; Evolutionary Algorithm; Genetic Algorithm; Crystal Structures; Titanium Dioxide.

*PACS:*61., 61.50.Ah, 61.66.-f

PROGRAM SUMMARY

Program Title: XtalOpt
Journal Reference:
Catalogue identifier:
Licensing provisions: GPL v2.1 or later [1]
Programming language: C++
Computer: PCs, workstations, or clusters
Operating system: Linux
Keywords: Structure Prediction; Evolutionary Algorithm; Genetic Algorithm; Crystal Structures; Titanium Dioxide.
Classification: 7.7
External routines/libraries: Qt[2], OpenBabel[3], Avogadro[4], and one of: VASP[5], PWSCF[6], GULP[7]
Subprograms used: SPGLIB[8]
Nature of problem: Predicting the crystal structure of a system from its stoichiometry alone remains a grand challenge in computational materials science, chemistry, and physics.
Solution method: Evolutionary algorithms are stochastic search techniques which use concepts from biological evolution in order to locate the global minimum on their potential energy surface. Our evolutionary algorithm, XtalOpt, is freely available to the scientific community for use and collaboration under the GNU Public License.
References:

[1] `http://www.gnu.org/licenses/gpl.html`
[2] `http://www.trolltech.com/`
[3] `http://openbabel.org/`
[4] `http://avogadro.openmolecules.net`
[5] `http://cms.mpi.univie.ac.at/vasp`
[6] `http://www.quantum-espresso.org`
[7] `https://www.ivec.org/gulp`
[8] `http://spglib.sourceforge.net`

*Corresponding author.
E-mail address: ezurek@buffalo.edu (E. Zurek)

1. Introduction

Over twenty years ago, John Maddox provocatively stated that the inability to predict the structure of a solid from its stoichiometry is a "scandal in the physical sciences," presenting a challenge to the chemistry, physics, and materials science communities [1]. Two decades later, crystal structure prediction remains exceedingly difficult. The difficulty stems from the complexity of the potential energy landscape of a solid, which depends on many variables: six unit cell parameters, plus three coordinates for each atom. Moreover, it may not be known how many atoms comprise the primitive unit cell. In order to determine the global (and ideally, all the local) minima, one requires a method which effectively samples the highly–dimensional parameter space of the extended system, while simultaneously focusing in on the most promising areas.

One way to predict structures is by using chemical intuition [2, 3]. However, in certain situations one may have insufficient empirical experience to make reasonable predictions. For example, consider the realm of high pressures, or compounds with unusual stoichiometries [4]. In these situations it may be prudent to resort to automated search techniques, the results of which can help to develop intuition in such unfamiliar circumstances. Performing local optimizations of randomly generated structures has shown success [5], but this technique is best suited to smaller systems since the probability of generating the global minimum decreases rapidly as the cell size grows. Simulated annealing [6], minima hopping [7], and metadynamics [8] are three promising search methods, but require a starting structure that is close to the target structure. They are very useful if the system is characterized reasonably well, but what if the structure is wholly unknown?

Genetic or evolutionary algorithms (GA/EAs) are stochastic search techniques that have been applied successfully in areas ranging from materials design and processing to molecular biology [10]. GAs/EAs have been extensively employed for predicting the structures of both molecules and clusters [11–17], as well as for extended systems [9, 18–28], surfaces [29], and even grain boundaries [30]. They draw upon concepts from biological evolution (i.e. survival of the fittest, crossover, trait propagation, mutation) to locate the global minimum of a system. These algorithms sample the multidimensional landscape of a system, while simultaneously focusing the search on the valleys. When applied to extended chemical systems, they require only the chemical composition as input (fulfilling Maddox's foremost request). The first successful application of this technique to crystal structures was performed by Woodley, Catlow, and Battle in 1995, in which the previously unknown structure of Li_3RuO_4 was discovered by their algorithm[18].

When applying a GA/EA to a specific problem, i.e. a crystal structure search, it is often illustrative to employ vocabulary taken from evolutionary theory. In the following sections we will refer to a particular candidate structure as an **individual** and the set of all structures generated in a single iteration of the algorithm as the **generation**. The union of all generations forms the **population**. The modification of an individual (a **parent**) to form a new structure is called **procreation**, and can occur via **mutations** (one parent) or **breeding** (two parents). The new structure is that parent's **offspring**. The subset of the population that is deemed fit for parenthood is referred to as the **breeding pool**, or just **pool**. Finally, each member of the population should occupy a unique **niche** to reduce the redundancy in the search.

Herein, a detailed description of XtalOpt, a new open-source evolutionary algorithm for crystal structure prediction, is provided. Computational details concerning licensing, implementation, compatible local optimizers, and GULP potentials used in the tests are found in Section 2. Details of XtalOpt's implementation and operation are provided in Section 3. This includes our choice of evolutionary operators, selection methods, workflow, niching techniques, and description of how to restrict the global minimization process to a reasonable parameter space. A unique "ripple" operator and XtalOpt's hybrid operator approach is presented in Section 3.2. Section 4 describes a newly proposed benchmarking technique and contains the results of a step-by-step parametrization of XtalOpt using a 16 formula unit TiO_2 supercell, as well as a set of validating tests performed on a more challenging $10\times SrTiO_3$ super cell. Results on NaH at normal pressure, and near the pressure induced structural phase transition are also provided. In Appendix A, one can find a brief tutorial of how to use XtalOpt.

2. Computational Details

2.1. Implementation

XtalOpt is written as an extension to the Avogadro[31] molecular editor and makes use of the OpenBabel[32, 33] C++ chemical toolkit. SPGLIB[34] is included for spacegroup identification. XtalOpt and its direct dependencies are available under the GNU Public License[35] (GPL), which allows free access to obtain, change, and improve the code. Users may also redistribute it as they see fit (the only restrictions imposed by the GPL is that modified versions be released under a similar license and that copyright information remains intact). The authors hope that the availability of this codebase will facilitate those interested in using and/or developing evolutionary algorithm approaches to the crystal structure problem. Furthermore, XtalOpt is interfaced with PWSCF, licensed under the GPL, and GULP, which is free for academic use. The VASP optimizer is also supported due to its popularity, but it is not free software. More details on these optimizers can be found in the next section.

The code for XtalOpt can be obtained free of charge or registration at http://xtalopt.openmolecules.net, or from the CPC library. It is written in C++ and uses TrollTech's Qt[36] libraries to provide the interface.

2.2. Supported Optimizers

XtalOpt currently supports three external codes that can be used to perform the local optimizations: VASP[37–40], PWSCF[41], and GULP[42–44]. The first two are first–principles, plane–wave programs, whereas the latter employs empirical potentials for discrete and extended systems. Examples of template inputs for each code are available on http://xtalopt.openmolecules.net and may also be obtained from the CPC Library.

2.3. Empirical Potentials

A 16 formula unit supercell of TiO_2 was chosen to parametrize XtalOpt in Section 4. This decision was based on the availability of an established and tested potential [27], the speed of the GULP optimizer, and historical use of TiO_2 as a test case for crystallographic evolutionary algorithms[19, 27].

The GULP potential for TiO_2 comes from Woodley and Catlow[27], where a combination of Buckingham and Lennard-Jones potential functions describe the interactions of the system such that

$$V(r) = Ae^{-\frac{r}{D}} + Br^{-12} - Cr^{-6}. \quad (1)$$

The parameters A, B, C, and D are dependent on the species in the interaction, and are provided in Table 1. This potential is combined with a Coulombic interaction term using charges of 2.196e for titanium and -1.098e for oxygen.

Table 1: Parameters used in the TiO_2 tests[a]

Species		A (eV)	B (Å^{12})	C (Å^6)	D (Å)
Ti	Ti	31120.1	1	5.25	0.1540
O	O	11782.7	1	30.22	0.2340
Ti	O	16957.5	1	12.59	0.1940

[a] Taken from Ref. 27

A validation test is performed on a 10 unit supercell of the ternary $SrTiO_3$ system using the same functional form as Equation 1 and the parameters in Table 2. The Coulombic interaction term is constructed using the formal charges of each species.

Table 2: Parameters used in the $SrTiO_3$ tests[a]

Species		A (eV)	B (Å^{12})	C (Å^6)	D (Å)
Sr	Sr	9949.1	1	0	0.2446
Sr	Ti	12708.1	1	0	0.2191
Sr	O	1805.2	1	0	0.3250
Ti	Ti	16963.1	1	0	0.1847
Ti	O	845.0	1	0	0.3770
O	O	22746.3	1	20.37	0.1490

[a] Modified from Ref. 45 in which B was taken to zero. Setting $B = 1$ separates atoms that are unphysically close to each other, without affecting the final geometry [27].

In both tests, optimization of the system was performed by reducing the norm of the gradient to 0.5 using a conjugate gradients approach, then switching to a BFGS minimization algorithm until GULP's default convergence criterion was reached. This approach ensured that most structures, including those far from the nearest minimum, converged successfully. All tests performed with GULP allow cell parameters to vary at a constant pressure of 0 GPa. Thus, the enthalpy is equivalent to the internal energy for the two test systems discussed herein. In addition computations on NaH at normal conditions, and near the pressure of the structural transition have been carried out using PWSCF.

3. Algorithm Details

3.1. Structural Representation

XtalOpt uses phenotypical representations, [9, 12, 13, 15–17, 20–29] meaning the operators responsible for procreation act directly on the actual coordinates and lattice parameters in either fractional or Cartesian space, as opposed to mutating and breeding the individual after encoding it as a binary string (or genotypical representation, analogous to a **chromosome** – examples of this approach also exist in the literature[11, 18, 19]). The use of phenotypical representations classifies XtalOpt as an evolutionary algorithm as opposed to a genetic algorithm.

3.2. Evolutionary Operators

When implementing an evolutionary algorithm, it is important to choose a set of operators that will explore the solution space thoroughly. In a crystal structure search, this requires investigation of the atomic positions, atomic ordering, and lattice parameters. There should also be a way for two individuals to communicate, or share information in the creation of new offspring. The evolutionary operators in XtalOpt fall into two categories: pure and hybrid operators. The pure operators each perform one of the aforementioned tasks, while the hybrid operators combine two pure operators into a single operation, with the goal of improving the search in some way (i.e. reducing the number of duplicate structures, or promoting exploration).

There are four pure operators in XtalOpt: crossover[9, 16, 20–23, 28] (providing communication), strain[21–23] (allowing variation of the lattice parameters), a newly proposed "ripple" operator (sampling new atomic positions), and exchange[9, 13, 21–23, 28] (exploring different atomic orderings).

The latter three pure operators are combined into two hybrid operators: "*stripple*", which merges strain with ripple, and "*permustrain*", which combines strain with exchange (also known as permutation [21]). These two hybrid operators, along with pure crossover, compose the three operators that our evolutionary algorithm uses to generate new structures. In Section 4 we describe the method employed to parametrize these operators and illustrate the effectiveness of the hybrid approach.

Crossover (Pure)

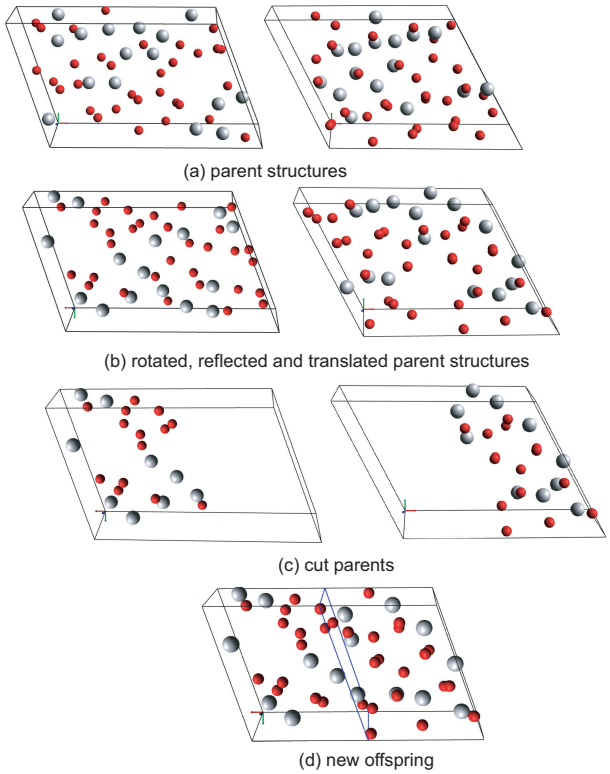


Figure 1: Example of the crossover operator, rendered in Cartesian coordinate space. For ease of visualization, parents with similar lattices are employed and the parents are not rotated during the transformation step (b).

The pure crossover (or "cut and splice") operator[9, 12, 16, 20–23, 28] combines two parent structures to form a single offspring. XtalOpt's implementation resembles the real-space mating operation of Deaven and Ho[12] that has more recently been adapted to crystals by Glass et al.[22] in their "heredity" operator. See Figure 1 for a stepwise example of how crossover works.

In our implementation, crossover begins by selecting two parents (Figure 1a). These parents are subjected to a series of random reflections and rotations to avoid biasing a given orientation, and all atoms are translated by the same random vector to vary the structural representation within the unit cell boundaries (Figure 1b). A spatially coherent subset of each parent's atoms are selected by making a "cut" perpendicular (in fractional coordinate space) to each transformed parent's **a** vector at the same random position (Figure 1c).

The offspring is formed by joining the atoms above the cut from one parent with the atoms below the cut from the other

parent. These "blocks" are joined in fractional coordinate space to avoid complications when aligning the cut ends of the Cartesian unit cell. The offspring's lattice dimensions are determined by taking a randomly weighted average of the parents' cell vectors. While it is in principle possible to make two children in this manner, XtalOpt creates only one, as illustrated in Figure 1. The child created is chosen randomly.

After the cut is performed, the number of atoms in the offspring is adjusted to match the input composition. If there are too many atoms of a given species, atoms of that type are randomly removed. If there are too few, atoms from the discarded portions of the parents are added. Figure 1d shows the resulting offspring.

The key distinctions between XtalOpt's crossover operator and others found in the literature[12, 22] are (a) the fractional coordinates of each parents' atoms are shifted along all three axes each time the operation is carried out, (b) the parents undergo reflections prior to cutting, and (c) a minimum contribution parameter is introduced to ensure that a significant portion of each parent is mixed into the offspring. This prevents the trivial mixing of structures, e.g. 1% of parent A and 99% of parent B.

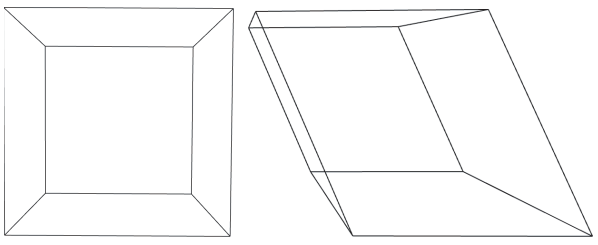
The crossover operator is configurable by setting the percentage of new offspring which it creates (p_c), and the minimum contribution of each parent ($p_{c,min}$).

Strain (Pure)

Pure strain is one of the most common single–parent operators employed for extended systems[21–23]. It multiplies each unit cell row vector $\mathbf{v}$ by the symmetric Voigt strain matrix:

$$\mathbf{v}_{\text{new}} = \mathbf{v} \begin{bmatrix} 1+\varepsilon_{11} & \frac{\varepsilon_{12}}{2} & \frac{\varepsilon_{13}}{2} \\ \frac{\varepsilon_{12}}{2} & 1+\varepsilon_{22} & \frac{\varepsilon_{23}}{2} \\ \frac{\varepsilon_{13}}{2} & \frac{\varepsilon_{23}}{2} & 1+\varepsilon_{33} \end{bmatrix} \quad (2)$$

where the ε_{ij} are random numbers taken from a zero–centered normal distribution with a specified standard deviation. The atomic coordinates are stored in fractional units during this operation in order to transform them along with the lattice. An illustration is provided in Figure 2.



(a) old unstrained lattice, simple cubic cell (b) new strained lattice, parallelepiped

Figure 2: An example of a unit cell (a) before, and (b) after the application of strain. Atoms have been removed from the lattice to ease visualization.

Ripple (Pure)

Our periodic displacement (or "ripple") operator shifts the coordinates of each atom along a random axis (for demonstration, we will choose the z axis) by an amount Δz, such that $z_{\text{new}} = z + \Delta z$. Acting in fractional coordinates, the displacement each atom undergoes depends on the atom's non-displaced (i.e. x and y) coordinates via the formula:

$$\Delta z = \rho \cos(2\pi\mu x + \theta_x) \cos(2\pi\eta y + \theta_y) \quad \mu \in \mathbb{Z},\ \eta \in \mathbb{Z} \quad (3)$$

Here, ρ specifies the maximum possible displacement of any atom in the displaced (z) direction, μ and η specify the number of cosine waves in each of the non-displaced lattice directions, and θ_x and θ_y are randomly chosen from the interval $0 \leq \theta < 2\pi$ to vary the regions of the structure that are strongly displaced. The name of this operator comes from the apparent "ripple" sent through the periodic lattice, as seen in Figure 3.

Examples of solids whose structures contain a ripple motif are found in nature. For example, various elemental high–pressure systems have modulated layer structures [46]: Cs–III [47], Rb–III [48] and Ga–II [49] can be seen as different stacking sequences of eight atom and ten atom layers. When viewed along the a axis, the atoms in the various layers appear to be placed approximately on the crests and troughs of a wave passing through the orthorhombic unit cell.

The nature of the directed coordinate mutation caused by the ripple operator allows regions of the individual to undergo little to no change, while distorting other regions considerably. This provides balance between allowing known good information to persist and exploration of the system's configuration space.

Exchange (Pure)

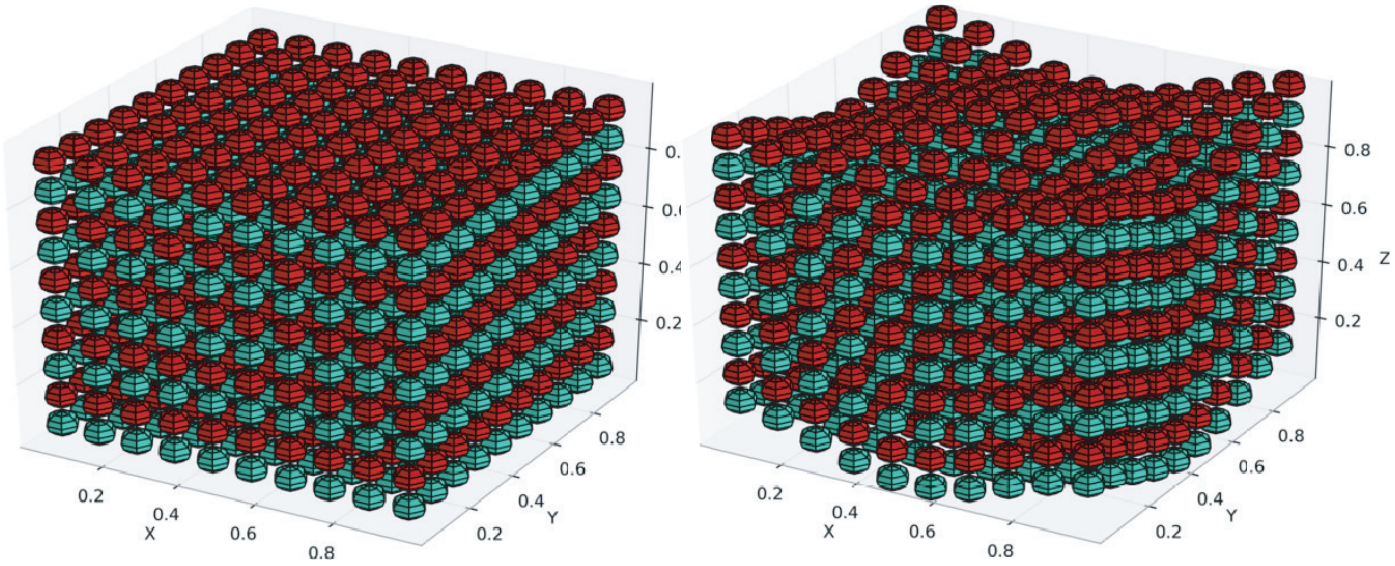
Exchange is another operator which has been extensively used for crystals[9, 13, 21, 22, 24]. It explores atomic ordering within the cell by exchanging the coordinates of two atoms of different types a specified number of times.

Stripple (Hybrid: Strain + Ripple)

The hybrid "stripple" operator combines strain with our chemically–motivated ripple operator. As shown in Section 4, both of these constituent pure operators show excellent performance when used alone, but they are prone to producing duplicate structures that waste computational resources by revisiting already–known minima. Through their unified efforts, however, the output of the hybrid stripple operator yields far fewer duplicates, meaning that more computational resources sample new regions of the potential energy landscape instead of re–examining known structures.

The percentage of new offspring formed with the stripple operator can be configured (p_s), and the degree of stripplation the parent undergoes is determined by the displacement amplitude range ($\rho_{s,min}$ and $\rho_{s,max}$), the periodicity of the cosine waves (η and μ), and the standard deviation range of the lattice strain ($\sigma_{s,min}$ and $\sigma_{s,max}$).

Prior to applying this operator, the input parameters $\rho_{s,max}$ and $\sigma_{s,max}$ are used as upper limits to assign a random value to the ρ_s and σ_s values used in the operations. As a result of this randomization, large maximum values will allow both major exploration of the potential energy surface and sampling of nearby configuration space during the course of the run. The



(a) before applying the ripple operator (b) after applying the ripple operator

Figure 3: An example of a 10×10×10 atom simple cubic structure (a) before, and (b) after applying the ripple transformation defined in Equation 3, $(\rho, \mu, \eta, \theta_x, \theta_y) = (0.08, 1, 1, 0, 0)$. To ease visualization, the ρ value is much lower than necessary for an effective transformation and the atoms are allowed to escape the unit cell boundaries.

corresponding minimum parameters are not hard limits, however. XtalOpt will guarantee that either $\rho_s \geq \rho_{s,min}$ or $\sigma_s \geq \sigma_{s,min}$, but not necessarily both. This allows either periodic displacement or strain to have a negligible effect. But both cannot, ensuring that the offspring will be significantly mutated compared to the parent.

The order in which the strain and displacement are applied is irrelevant, as each acts on independent aspects of the structure (the lattice and fractional atomic coordinates, respectively).

Permustrain (Hybrid: Exchange + Strain)

Exchange (also known as permutation [21])is combined with strain to form the "permustrain" operator, which explores lattice configurations while varying atomic ordering.

The percentage of new structures formed using permustrain is configurable (p_p), as well as a maximum strain standard deviation ($\sigma_{p,max}$) and the number of independent atom exchanges (N_{ex}). When applying permustrain, the actual σ_p value will be randomized over the range $0 \leq \sigma_p \leq \sigma_{p,max}$. By allowing σ_p to be near-zero, it is possible for pure exchange to occur.

Like strippling, the two constituent operations commute so that the order of applying strain and permutation does not affect the offspring structure. By itself, exchange leads to few duplicate structures. Nonetheless, combining these two operators also results in a decrease of the number of duplicates obtained from applying strain or exchange alone. More importantly, combining these pure operators hastens exploration of the potential energy surface. Section 4 provides results to support both claims.

3.3. Selection

Apart from evolutionary operators, another defining feature of an EA is the fitness function, which provides a measure of an individual's worth and allows a determination of which individuals are best suited to become parents. Most GAs/EAs employed for chemical structure prediction use a thermodynamic quantity such as the individual's energy, enthalpy, or free energy to determine its fitness[9, 11–13, 15, 16, 20, 22, 23, 27–29]. XtalOpt is no exception, and uses the enthalpy ($H = U + PV$) as the quantity by which to assess the stability of an individual. This allows XtalOpt to be used on solids under high pressure, but we point out that our implementation is by no means limited to such searches. The pressure may be specified by supplying the appropriate information in the input template for the external optimizer. This will not affect the behavior of XtalOpt.

The probability p_i of selecting a given individual for breeding is calculated from its enthalpy H_i as:

$$p_i = N\left(1 - \frac{H_i - H_{min}}{H_{max} - H_{min}}\right), \tag{4}$$

where H_{min} and H_{max} are the enthalpies of the best and worst individuals in the pool and N is a normalization constant ensuring that $\sum p_i = 1$.

3.4. Continuous structure generation

In both traditional Lamarckian evolutionary algorithms and XtalOpt, the initial population of structures is either randomly generated or seeded by the user. These then undergo time–consuming structural optimization. Traditional algorithms wait for all of the local optimizations to finish before evaluating the fitness of the optimized structures and constructing the next

generation of unoptimized structures via procreation. The new set of structures are sent for optimization, and the process is repeated until an endpoint is reached. If a single individual is slow to optimize, this generation–at–once breeding frenzy creates a bottleneck in the algorithm, preventing new structures from being generated for hours or even days.

To alleviate this problem, Bandow and Hartke introduced a continuous mode of operation in their studies of water clusters [15]. This approach differs from a traditional evolutionary algorithm by procreating a new offspring for minimization immediately every time an individual optimizes, rather than waiting for an entire generation to complete. This procedure takes full advantage of parallel processing, since a predefined number of structural optimizations are running constantly, eliminating the bottleneck described above. The parents of new offspring are chosen from a pool of the fittest individuals in the entire population (a population–based pool [15, 29]) as opposed to selecting from only the previous generation (a generation–based pool). Bandow and Hartke[15] have clearly documented the more efficient usage of computational resources in a continuous mode compared to traditional generation–based algorithms.

XtalOpt implements this workflow for its structural optimizations. The concept of "generation" is refined in XtalOpt's continuous mode to denote the number of individuals that separate an individual from the inital seeded or random structures; generations hold only a trivial position in this mode of operation.

Care must be exercised when using a population-based selection pool to ensure that duplicate individuals are found and removed from the population. Since the individuals are chosen by enthalpy-weighted probabilities, any duplicate structure in the pool will unfairly increase that individual's chances of breeding. To reduce this problem, XtalOpt uses a "niching" strategy, outlined below.

3.5. Niching

The removal of duplicate individuals, or niching, is essential to prevent stagnation of the breeding pool, or becoming "stuck" on a single structure. Previous implementations of niching have relied on comparing the energy or enthalpy of a structure[12, 24], which works fine for most systems but can produce false positives if two different structures have similar enthalpies. For clusters, Li et. al. have employed a geometric criterion in which bond distances are compared [50]. Abraham and Probert[26] proposed an advanced niching technique to examine extended systems, however we have opted for a simpler method that compares three inherent properties of the structure (the individual's **fingerprint**) in order to eliminate redundancies. Others have defined fingerprint functions to determine the similarity between various structures [51]. However, these have been employed primarily as post–analysis techniques in order to quantify energy landscapes, and not as niching schemes.

The components of a fingerprint in XtalOpt are the enthalpy, spacegroup, and volume of the structure. The enthalpy and volume are checked to be within a specified tolerance, and the spacegroups of the structures are found using Dr. Atsushi Togo's SPGLIB library [34] and compared. If the fingerprints of the two individuals match, the least favorable individual (there is usually a small enthalpy difference) is **killed** and removed from the population, leaving the other to continue. In case that the enthalpies match exactly, an arbitrary structure is killed.

This method of determining duplicates may lead to false positives, or negatives in rare situations. For example, it is possible for two different structures to have the same fingerprint according to the niching scheme described above, especially if the spacegroup has little or no symmetry (ie. $P1$ or $P\bar{1}$). Moreover, a certain spacegroup may not be recognized because the atomic coordinates differ slightly from their ideal positions.

3.6. Restraints

In order for an EA to be efficient, it must be restricted to a physically reasonable search space. Such "directed" evolution is necessary to take into consideration various environmental factors. For example, under sufficient pressure the PV term becomes the dominant contribution to the enthalpy ($H = U + PV$), and cells with too large of a volume will be highly destabilized. Thus, it is possible to accelerate the search by focusing on cells with a specified volume.

In XtalOpt various restraints may be employed to direct the search, depending upon how much is known about the target structure. For example, the lattice parameters and volume may be constrained to a range or be fixed. Other restrictions allow one to limit the minimum interatomic distances to prevent overlap of atoms. Of course, if nothing other than the composition is known about the system a loose set of limits may be used, although it is beneficial to provide reasonable limits guided by chemical intuition, to ensure a well-behaved search. For example, minimum interatomic distances may be estimated using the Shannon radii [52]. All restrictions are enforced before local optimizations are performed, and by default the structures are permitted to optimize outside of the specified values.

In addition, the structure's lattice is adjusted prior to optimization so that all angles are between 60° and 120° by a process similar to that described by Oganov and Glass[25]. The following transformations are applied to all combinations of the three lattice vectors:

If both

$$\left|\arccos\left(\frac{\mathbf{v_i}\cdot\mathbf{v_j}}{\|\mathbf{v_i}\|\,\|\mathbf{v_j}\|}\right)-\pi\right| > \frac{\pi}{3}, \quad \text{and} \tag{5a}$$

$$\|\mathbf{v_i}\| \geq \|\mathbf{v_j}\| \tag{5b}$$

are true, the cell should be transformed. To do this, the vector $\mathbf{v_i}$ is replaced by the vector $\mathbf{v_i'}$, defined as

$$\mathbf{v_i'} = \mathbf{v_i} - \mathrm{ceil}\left(\left|\frac{\mathbf{v_i}\cdot\mathbf{v_j}}{\|\mathbf{v_j}\|^2}\right|\right)\mathrm{sign}(\mathbf{v_i}\cdot\mathbf{v_j})\mathbf{v_j} \tag{6}$$

Atomic coordinates are stored in Cartesian form before the lattice transformation to maintain the structure's atomic configuration. After the angles are set, the atoms are adjusted so that they lie within the new cell by adding or subtracting 1 from the new fractional coordinates.

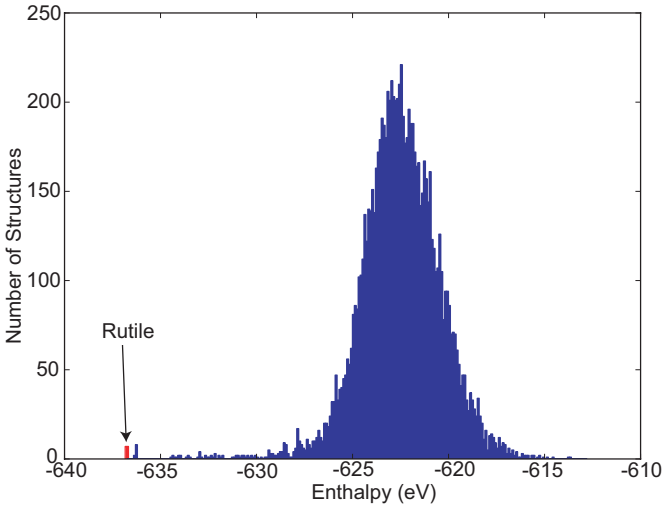


Figure 4: Distribution of the enthalpies of randomly generated structures containing 16 titanium and 32 oxygen atoms after local optimization has been performed. Out of 9943 structures, only 7 corresponded to the global minimum of TiO_2, rutile.

Eqs. 5a, 5b, and 6 differ somewhat from those provided by Oganov and Glass[25], since the above equations apply to vectors of arbitrary length. A graphical example of this transformation with a detailed explanation is available online at http://xtalopt.openmolecules.net.

4. Results

4.1. Quantifying the Performance of an Evolutionary Algorithm

Various polymorphs of TiO_2 are known (anatase, brookite, hollandite, and ramsdellite), however rutile is the global minimum at normal pressures. In order to provide reasonable settings for various parameters in our EA rutile was used as a test case in the benchmarking procedure, detailed below. But first we wanted to find out how well a random structure search fares when presented with this problem.

9943 cells containing 16 TiO_2 units were randomly generated and then optimized using GULP and the potentials provided in Section 2.3. Figure 4 shows the distribution of enthalpies obtained from the randomly constructed atomic configurations. Of these structures, only seven (0.074%) produced the rutile supercell, implying that ~1450 random structures must be generated in order to have some assurance that the global minimum would be found. Can an evolutionary algorithm do better than this? Shortly we will show that for this system an EA (employing the proper parameters) is consistently able to find rutile in fewer than 600 structures, demonstrating that yes indeed it can.

One particularly difficult aspect of characterizing an evolutionary algorithm is overcoming the stochastic nature of the method. There is always a probability that the target structure may appear in the random first generation, or not at all in a given search (See Figure 5). A common technique is to report the average number of generations or structures that it takes a search to produce the target structure[9, 12, 19, 23, 24, 29], but this method is problematic – without using an enormous number of structures in each trial, it is likely that at least some of the tests will not find the global minimum.

Therein lies the dilemma: how to account for the failed searches in the average? These numbers are highly system specific, and one can expect the estimates to change drastically with cell size and the number of different atom types. To avoid this problem, we propose a method that quantifies the evolutionary algorithm's performance using a modest number of structures per trial. Our method doesn't require each trial to find the global minimum, facilitating fast, automated benchmarking. For example, in the following sections the results of 45 tests, each consisting of 100 searches apiece — a total of 4,500 independent searches using the EA — are provided. A benchmarking technique that does not require manual intervention or a guarantee of success is required for such an undertaking.

The analysis proposed herein is an extension of the technique introduced by Hartke, which takes data from several searches ($N = O(100)$) with identical parameters and considers the individual with the lowest energy in each generation [11]. The original "Hartke plot" shows three energy trends on the same plot: the worst search's best individual by generation, the average best individual by generation, and the best search's best individual by generation. Of particular interest to us is the "average best" plot, which shows a curve resembling a standard exponential decay. In Figure 5 an example of a "Hartke plot" obtained during our parametrization of XtalOpt is provided. Since XtalOpt employs a continuous mode, the x-axis corresponds to the number of structures generated, rather than the number of generations considered.

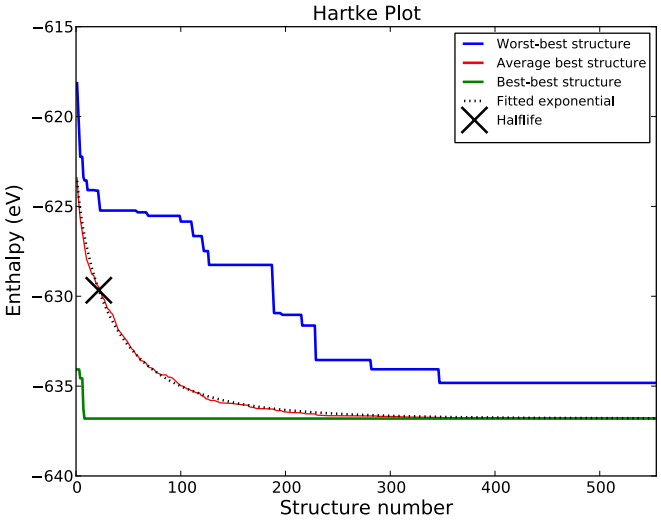


Figure 5: An example of a "Hartke plot" with the best, average, and worst enthalpy runs plotted using solid lines. The dotted line shows the fit described by Equation (7) and a "×" denotes the half–life point. It can be seen that some searches never produced the target structure rutile (blue line, top), while others found rutile in the first few structures (green line, bottom), illustrating why a large number of searches are needed to benchmark a given parameter set.

We have taken Hartke's idea a step further and fit a function

of the form

$$\bar{H}(i) = (\bar{H}_{\text{rand}} - H_{\text{min}})e^{ai^b} + H_{\text{min}} \quad (7)$$

to the average enthalpy curve. In this equation, a and b are parameters to be determined, i is the structure index, and H_{min} is the enthalpy of the most stable (target) individual. $\bar{H}_{\text{rand}}$ is the average enthalpy of a large number ($N = O(10,000)$) of locally optimized random structures. This choice of constants removes variations from the endpoints by forcing $\bar{H}(0) = \bar{H}_{\text{rand}}$ and $\lim_{i\to\infty}(\bar{H}(i)) = H_{\text{min}}$. All structures from the first (random) generation are removed before fitting so that the structure at $i = 1$ is the first individual generated via an evolutionary operator.

To quantify the fitness of a given parameter set, many (~100) searches are performed using a test system, and the Hartke plot is generated along with the $\bar{H}(i)$ fit. The efficiency is related to the half–life of the exponential curve, $i_{\frac{1}{2}}$, which represents how many individuals it takes for $\bar{H}(i)$ to drop to an enthalpy halfway between a locally optimized random structure and the global minimum. Thus, lower $i_{\frac{1}{2}}$ values indicate that an average search approaches the minimum more quickly. However, this half–life value is not the only metric considered. The percentage of searches that find the global minimum by a predefined structure number (% success) and the number of duplicate structures produced during the run (% duplicates) as determined using the previously described niching strategy are used to obtain a clear idea of the performance of a particular parameter set.

Individually, each of these measures provides insight into a search's performance. A steep decline along the $\bar{H}(i)$ curve, or a low half–life, suggests that the search is quickly locating the valleys and exploring them. A high success rate obviously indicates that a search is performing well, but a low success rate can offer clues as to what is happening. For example, if a parameter set yields a good (low) half-life but a bad (low) success rate, this suggests that the algorithm has difficulty escaping metastable local minima — it quickly finds reasonable structures, but fails to further improve upon them. The percentage of duplicates shows how effectively the computational resources are being used: a high number of duplicates means that cycles are being wasted on re-exploring known minima. The number of duplicate structures can also be used as an indication of the diversity maintained during a search.

It should be pointed out that each of these metrics has an error margin associated with it. From examining our tests on TiO_2, we estimate the uncertainty to be ±5 structures for the half-lives, ±3% for the success rates, and ±1% for duplicate rates. This is the reason why, for example, the entries in the fourth row in Table 6 ($N_{\text{ex}} = 4$) differ slightly from those of the first row in Table 8 ($\sigma_{\text{p,max}} = 0$), even though these searches had essentially identical parameters.

4.2. Parametrization with TiO_2

XtalOpt has been parametrized in three steps. First, tests on TiO_2 were carried out to determine the best parameters for the pure operators alone. These operators were then mixed, and the hybrid operators were tested to find the settings that maximized their performance. Finally, the percentage of new structures that each operator should generate during a production run was determined. Additionally, a validation test using a more challenging ternary oxide system, $SrTiO_3$, was performed.

Each test consisted of 100 unique searches, with 600 structures generated during the course of each search. We considered a system with 32 oxygen and 16 titanium atoms, targeting rutile as the global minimum. The randomly generated initial population had 20 structures. The restraints used were loose – cell lengths were limited from 0 to 40 Å, and cell angles were limited to 60°-120°. The volumes of all structures were initially scaled to 485 Å^3. These same restraints were used in the random search (Figure 4) as well. In the fit according to Equation 7, H_{min} was the enthalpy of sixteen rutile unit cells (-636.806 eV), and $\bar{H}_{\text{rand}}$ was the average enthalpy of the randomly generated set of structures from Figure 4 (-622.528 eV). The least–squares method in Python's SciPy[53] library was employed to determine a, b during the fitting procedure, and a custom Newton–Raphson routine was used to calculate $i_{\frac{1}{2}}$. The volume (tolerance of 0.002 Å^3), enthalpy (tolerance of 0.002 eV), and spacegroup were used for niching.

Individual Operators

Starting with the pure crossover operator, we varied the minimum amount a parent could contribute to the child structure, $p_{\text{c,min}}$, and monitored the aforementioned metrics. As Table 3 shows, crossover led to the creation of very few duplicate structures, indicating that this operator samples the potential energy surface well. However, the half-life was the highest of any operator which was considered, and the likelihood of finding rutile in an average run consisting of 600 structures could be improved.

Table 3: The influence of the minimum contribution parameter, $p_{\text{c,min}}$, on the efficiency of the crossover operator. The half-life ($i_{\frac{1}{2}}$), the percentages of searches that found rutile (% success), and the % duplicates that were found are compared.

$p_{\text{c,min}}$ (%)	$i_{\frac{1}{2}}$	% success	% duplicate
0	26.9	97.0	0.7
10	40.5	91.0	0.2
20	30.6	80.0	0.2
30	35.2	85.0	0.1
40	37.5	86.0	0.1
50	36.1	86.0	0.1

Interestingly, setting $p_{\text{c,min}} = 0$ resulted in the lowest half-life and the highest success rate in this set of runs. This setting allows for the probability that the child structure consists ~100% of parent A, and ~0% of parent B. Since the lattice vectors of the child are a randomly weighted average of those of the two parents, it is highly unlikely that the child would be equivalent to parent A. More likely, the cell shape of parent A would be modified. Thus, this setting allows crossover to behave in a similar fashion as the strain operator.

Comparison with Table 4 shows the performance of the pure

strain operator is far superior to that of crossover. Thus, the lower $i_{\frac{1}{2}}$ and higher success rate for $p_{c,min} = 0$ can be attributed to the strain–like behavior that results from trivial mixing. Since the goal of crossover is to provide communication between two structures (a hallmark of a GA/EA that does not occur in these "pseudo–strain" crossovers), we do not consider the $p_{c,min} = 0$ data when proposing a value for $p_{c,min}$. It appears that a $p_{c,min}$ of between 20-30 would provide a balance between a lower half–life, and a higher success rate, while at the same time not being too large.

In Table 4 the same metrics for varying the minimum standard deviation, σ_{min}, of the pure strain operator are provided. Strain has a much lower half–life than crossover, and superior success rate: each test was certain to find rutile within the first 600 structures generated. The drawback of strain is that it suffers from a high percentage of duplicate structures. The number of duplicates decreases as σ_{min} increases, suggesting that if a small strain is applied, the structure will revert to the original after local optimization. It appears that a minimum strain value of 0.5 or greater lessens the likelihood of these ineffective mutations.

Table 4: Same as Table 3, except for the strain operator, and varying the minimum standard deviation of the lattice strain, σ_{min}.

σ range	$i_{\frac{1}{2}}$	% success	% duplicate
0.0-1.0	15.8	100.0	26.4
0.1-1.0	18.0	100.0	21.8
0.2-1.0	15.9	100.0	14.6
0.3-1.0	17.6	100.0	10.0
0.4-1.0	15.7	100.0	8.3
0.5-1.0	14.2	100.0	6.2

Before combining strain with the ripple operator to form stripple, we needed to find out what minimum ripple amplitude (ρ_{min}) would be necessary to reduce the number of duplicates to a reasonable level. As Table 5 shows, the success rate and half–life of ripple was comparable to that of strain. Once again, the number of duplicates was quite high and decreased with increasing ρ_{min}, confirming that structures which were not sufficiently modified by the mutation had optimized back to the original parent. In order to keep diversity high in the structure search, a minimum amplitude of at least 0.5 is suggested.

Table 5: Same as Table 3, except for the ripple operator, and varying the minimum amplitude, ρ_{min}, of the ripple sent through the cell.

ρ range	$i_{\frac{1}{2}}$	% success	% duplicate
0.0-1.0	17.6	99.0	22.4
0.1-1.0	16.5	97.0	17.6
0.2-1.0	19.2	98.0	12.2
0.3-1.0	15.5	99.0	9.0
0.4-1.0	15.9	99.0	7.1
0.5-1.0	16.4	100.0	7.3

Although the half–life of exchange was only slightly higher than that of the strain or ripple operators, the success rate was markedly lower (Table 6). Swapping a small number of atoms led to a high percentage of duplicates. If exchange is used by itself, at least three atoms must be interchanged in order to maintain diversity in the offspring. However, swapping too many atoms leads to a loss of good structural information. The ideal setting for this variable will undoubtedly depend on the number of atoms and atom types in the unit cell. For cells containing around 50 atoms, $N_{ex} = 3, 4$ seems to be a good balance which will allow for a diverse population, while retaining information.

Table 6: Same as Table 3, except for exchange, and varying the number of atom swaps, N_{ex}.

N_{ex}	$i_{\frac{1}{2}}$	% success	% duplicate
1	12.0	86.0	36.4
2	17.1	94.0	13.3
3	19.2	85.0	7.2
4	21.1	92.0	3.5
5	22.9	69.0	1.9
6	21.0	89.0	1.2

The aforementioned tests indicate that strain and ripple do quite well on their own. Can the hybrid stripple outperform them in some way? As Table 7 shows, merging the two operators into one significantly reduced the number of duplicates. Moreover, the searches nearly guaranteed success, and the best half–lives were comparable to those of the pure operators. Thus, these tests indicate that strippling is superior to pure strain and ripple, since it does a better job of sampling the configuration space, while still being able to zoom in on the most stable structure.

The $\sigma_{s,min}$ and $\rho_{s,min}$ have been set to the values determined for strain and ripple alone. These minimum values are not hard limits, and only one of them is enforced, allowing either strain or ripple to be applied with negligible effect. Setting $\sigma_{s,max} = 0.5$ and $\rho_{s,max} = 1$ resulted in the most efficient exploration of the potential energy surface, with a balance between low half-life and few duplicates.

Table 7: Same as Table 3, except for the hybrid stripple operator. Different combinations of $\rho_{s,max}$ and $\sigma_{s,max}$ values were tested. The minimum values used were $\rho_{s,min} = \sigma_{s,min} = 0.5$.

$\rho_{s,max}, \sigma_{s,max}$	$i_{\frac{1}{2}}$	% success	% duplicate
0.5,0.5	17.0	100.0	4.9
0.5,1.0	20.1	99.0	2.6
0.75,0.75	24.3	99.0	0.8
1.0,0.5	17.9	100.0	2.7
1.0,1.0	32.0	100.0	0.4

To allow pure exchange to occur with a finite probability, the hybrid permustrain operator does not enforce a minimum strain standard deviation, allowing σ_p to be small. The number of exchanges, N_{ex}, was set to four (ensuring that for a pure swap the number of duplicates would remain low), and the maximum deviation of the strain, $\sigma_{p,max}$ was varied. As Table 8 shows, the success rates and half-lives of the hybrid are comparable to those of exchange alone. However, the number of dupli-

cate structures could be decreased by varying $\sigma_{\mathrm{p,max}}$. Setting $\sigma_{\mathrm{p,max}} = 0.4 - 0.6$ ensures a low duplicate count, without significantly affecting the success rate or half-life as compared to pure exchange.

Table 8: Same as Table 3, except for the hybrid permustrain operator. Different values of $\sigma_{\mathrm{p,max}}$ were tested with N_{ex}=4 swaps.

$\sigma_{\mathrm{p,max}}$	$i_{\frac{1}{2}}$	% success	% duplicate
0.0	19.6	89.0	3.9
0.2	20.5	89.0	3.5
0.4	23.2	85.0	1.7
0.6	23.9	90.0	0.8
0.8	25.3	88.0	0.5
1.0	28.4	92.0	0.6

Combining the Operators

Now that a reasonable set of values for all of the parameters in the individual operators has been found, an investigation of which ratios of crossover (p_{c}), stripple (p_{s}), and permustrain (p_{p}) give rise to the most fruitful search was carried out. All parameters were set to the values determined above (also listed in Table 10). The percentage of each operator has been varied, and the results are given in Table 9. The first seven rows show that all of our runs resulted in a small number of duplicates, and most found rutile within 600 structures. Setting $(p_{\mathrm{c}}, p_{\mathrm{s}}, p_{\mathrm{p}})$=$(15, 50, 35)$% gave the lowest half–life. However, due to the estimated error of ±5 structures associated with $i_{\frac{1}{2}}$, one can only conclude that all of the ratios considered appear to give a reasonably good search. It is comforting to see that the evolutionary runs outperformed the random search (see Figure 4) which needs an estimated 1450 structures on average to find the rutile supercell.

Table 9: Same as Table 3, except for varying the ratios of the number of structures created using the crossover (p_{c}), stripple (p_{s}), and permustrain (p_{p}) operators. For each of these, the parameters given in Table 10 were employed. For comparison, another parameter set published in the literature has also been tested[a].

$p_{\mathrm{c}}, p_{\mathrm{s}}, p_{\mathrm{p}}$	$i_{\frac{1}{2}}$	% success	% duplicate
15,40,45	21.4	97.0	1.5
15,50,35	17.0	100.0	1.7
25,30,45	20.3	100.0	1.2
25,40,35	21.7	98.0	1.6
25,50,25	20.6	100.0	1.5
35,30,35	23.1	99.0	1.2
35,40,25	21.7	100.0	1.4
85,15,5[a]	25.2	99.0	1.1

[a] The parameters were chosen to resemble those used in Ref. 22 as closely as possible.

Unfortunately, it is not common practice in the literature to state all of the parameters used in an evolutionary search, making direct comparison to other algorithms difficult. However, Glass et al. give a particularly good description of their operators and parameters in Ref. 22. The authors have suggested to use 85% crossover, 15% strain and 5% exchange. In the first seven rows of Table 9, the percent of crossover was kept relatively low, since our previous computations showed that this operator alone gave a significantly higher half-life than permustrain or strippling. We wondered how well a setting with a high p_{c} would fare, and decided to test out the parameters suggested in Ref. 22.

It is important to note that our method differs from the algorithm described in Ref. 22 — for example, we use a continuous mode of operation with a population based pool and apply a niching scheme. Our operators differ as well, but by supplying proper parameters the operators described by Glass et al. can be emulated. In performing this comparison, the minimum crossover contribution was set to zero. The stripple operator was fixed to apply no ripple and always use a strain standard deviation of 0.7. Permustrain used three exchanges with no strain. The operator ratios were set to $(p_{\mathrm{c}}, p_{\mathrm{s}}, p_{\mathrm{p}})$=$(85,15,5)$%, and all other values to those in Table 10.

As the last row of Table 9 shows, even taking the error inherent in $i_{\frac{1}{2}}$ into account, the half–life obtained from the parameter set suggested in Ref. 22 is larger than our best set of values. Most likely this is due to the higher p_{c} employed, however it is not quite clear since the computations varied in a number of respects, as described above. The success rates, and number of duplicate structures from all of the runs are comparable.

4.3. A More Difficult Test: $SrTiO_3$

Based upon the tests carried out on the TiO_2 supercell, we recommend the values in Table 10 for a generic XtalOpt search. While evolutionary algorithms are system specific and these parameters will not be ideal for all cell compositions, they serve as a starting point that should allow most searches on binary systems with moderately sized unit cells (less than 50 atoms) to find the global minimum within the first few hundred structures. We have not specifically tested the effect of varying the size of the first generation (N_i), nor the size of the breeding pool (N_{pool}). Both should be large enough so as to allow for diversity in the breeding pool, but not too large so as to waste computational time. The energy/enthalpy and volume niching tolerances will also be system specific, and depend upon the precision settings requested in the computation.

Finally, we wanted to see how this set of parameters would perform on an even more challenging system. A 10 unit supercell of the ternary $SrTiO_3$ (50 atoms total), which crystalizes in the perovskite structure was chosen.

In order to determine $\bar{H}_{\mathrm{rand}}$ for the fit in Equation 7, a random search consisting of 100,391 structures was carried out. The perovskite structure was not found even once, indicating that this is a much more demanding system. The enthalpy of the best randomly generated structure was actually 0.11 eV/atom higher than that of the global minimum.

In these tests, the parameter set provided in Table 10 was employed, along with the GULP optimizer, and the potentials given in Section 2.3. The results were compared with those obtained when our algorithm was modified so that it replicated the method described in Ref. 22 as closely as possible. Both

Table 10: Suggested default parameters for a generic XtalOpt search.

Parameter	Description	Value
N_i	Size of first generation	20+
N_{pool}	Size of breeding pool	15+
p_{c}	Percentage of new structures generated via crossover	15
$p_{\text{c,min}}$	Minimum contribution of each parent	25
p_{s}	Percentage of new structures generated via stripple	50
$\sigma_{\text{s,min}}$	Minimum standard deviation of ε (Eq. 2)	0.5
$\sigma_{\text{s,max}}$	Maximum standard deviation of ε (Eq. 2)	0.5
$\rho_{\text{s,min}}$	Minimum amplitude of periodic displacement (Eq. 3)	0.5
$\rho_{\text{s,max}}$	Maximum amplitude of periodic displacement (Eq. 3)	1.0
η, μ	Number of "waves" per cell (Eq. 3)	1,1
p_{p}	Percentage of new structures generated via permustrain	35
$\sigma_{\text{p,max}}$	Maximum standard deviation of ε (Eq. (2)	0.5
N_{ex}	Number of atom swaps	3-4
ΔH	Tolerance for enthalpy niching	~ 0.04 meV/atom
ΔV	Tolerance for volume niching	$\sim 1 \times 10^{-4}$ Å^3/atom

tests were performed using 100 independent searches that terminated after 1000 structures had been generated. The values used in fitting Equation 7 were $\bar{H}_{\text{rand}} = -1484.522$ eV and $H_{\text{min}} = -1500.312$ eV.

Table 11: The half–life, percentage of successful searches, and percentage of duplicates obtained when using the parameter set from Table 10, and a ratio of $(p_{\text{c}}, p_{\text{s}}, p_{\text{p}})$=(15, 50, 35)%. We have also considered another parameter set published in the literature[a]. The target was the perovskite structure in a 10× unit cell of $SrTiO_3$.

$p_{\text{c}}, p_{\text{s}}, p_{\text{p}}$	$i_{\frac{1}{2}}$	% success	% duplicate
15,50,35	566.3	12.0	2.2
85,15,5[a]	1460.9	7.0	1.0

[a] The parameters were chosen to resemble those used in Ref. 22 as closely as possible.

As Table 11 illustrates, the ternary perovskite was indeed a more difficult system than rutile, despite the fact that the two had roughly the same number of atoms in the unit cell. While the success rates for the TiO_2 supercell were nearly perfect in 600-structure searches, only 7-12% of the 1000-structure searches find perovskite $SrTiO_3$. Nonetheless, it was encouraging to see that our success rate was nearly double, and our half–life was about 2.5 times lower than one obtained using parameters similar to those proposed in Ref. 22. This test indicates that the parameter set proposed in Table 10 will likely provide reasonable results for different systems.

However, this data illustrates that there is still room for improving crystallographic evolutionary algorithms. For example, we are currently considering new ways by which communication between two structures can occur, so as to improve the half–lives and success rates obtained by a two–parent operation.

4.4. Tests with Plane–Wave DFT Codes

Due to the computational expense involved, we were not able to carry out tests similar to those outlined above for TiO_2 and $SrTiO_3$ using a plane–wave DFT code. Systems for which thousands of optimizations could be performed in a reasonable amount of time using a first–principles program contain only a few atoms in the unit cell. In these cases, the global minimum is found quickly, perhaps even in the first set of randomly generated structures. The performance of XtalOpt with both PWSCF and VASP has been tested. Unfortunately, because of the reasons listed above we are only able to report the results of a single search, and not of a series of benchmarks.

At normal pressure all of the alkali hydrides crystallize in the rocksalt structure with one atom of each type in the primitive unit cell. Experiments have shown that when compressed these ionic hydrides (except for LiH) transform to the CsCl structure [54]. For NaH, this transition has been found to occur at 29.3 GPa [55]. Our PWSCF computations predict the transition to occur just under 35 GPa. We have employed XtalOpt to find the most stable structure of a quadruple unit cell (8 atoms in total) of NaH at normal pressure. In the PWSCF run the rocksalt structure was found quickly, and almost all of the systems generated were the global minimum. A more interesting run where the pressure was set to 29.3 GPa in the PWSCF input file was also performed. According to our calculations, at this pressure the rocksalt structure is only 17 meV/atom more stable than the CsCl configuration. Out of all of the structures which were generated by XtalOpt, about 16 % were the global minimum, whereas approximately 75 % had the slightly less stable CsCl structure, and the rest were even less stable. Similar results have been obtained with VASP. We are currently using XtalOpt to predict the structures of solids under pressure since these may have exotic structures and/or stoichiometries [2–5].

5. Conclusion

Herein, we outline an implementation of an evolutionary algorithm, XtalOpt, which may be used for predicting crystal structures. Our method has been interfaced with various plane–wave density functional codes (VASP, PWSCF), as well as GULP, which employs empirical potentials. Importantly,

XtalOpt is published under the GNU Public License, and it is freely available at http://xtalopt.openmolecules.net or from the CPC library. XtalOpt has been written as an extension to the Avogadro molecular editor. It has an intuitive, easy to use graphical interface, with numerous useful data analysis features. A brief tutorial of XtalOpt, which explores its interface and selected features is given in Appendix A.

A chemically motivated ripple operator, which sends a wave through the crystal, has been introduced. Moreover, it is shown that combining two pure operators into a hybrid (ie. stripple which consists of strain and ripple, or permustrain which is made up of permutation and strain), results in superior performance by decreasing the number of duplicate structures. This is essential for our algorithm, since it uses a continuous workflow and a population based pool. In order to keep the diversity in the pool high, a niching scheme which compares the volumes, enthalpies and spacegroups of the structures is employed.

Suggested values for various parameters in XtalOpt have been set by performing benchmarks on a $16\times TiO_2$ supercell of rutile. We have introduced a way to quantify the performance of an evolutionary algorithm by fitting an exponential decay function to the average best structure data obtained in a Hartke plot and calculating the half–life. This gives a metric of how quickly an average search will approach the global minimum. Other metrics, such as the success rate, and number of duplicate structures were also employed to determine the efficiency of our algorithm. Tests on a more difficult $SrTiO_3$ supercell indicate that XtalOpt performs much better than a random structure search. Nonetheless, various operators and parameters could potentially be improved.

We expect that chemists, physicists, and materials scientists interested in predicting the structures of solids will find XtalOpt to be a useful tool towards this end.

Acknowledgments

The authors thank Dr. Pio Baettig for his testing and feedback. We thank Dr. Geoffrey Hutchison for hosting XtalOpt at http://openmolecules.net, as well as for guidance concerning integration with Avogadro. Dr. Hutchison and Dr. Marcus Hanwell (as well as the rest of the Avogadro team) frequently provided technical insight necessary for much of the integration work. Dr. Atsushi Togo, author of the SPGLIB spacegroup identification library, was extremely helpful with interfacing XtalOpt with his code, and the authors are grateful for his dedication to providing a sound, functional code. We acknowledge support from the Center of Computational Research at SUNY Buffalo, and the NSF through TeraGrid resources provided by the National Center for Supercomputing Applications.

Appendix A. Brief XtalOpt Tutorial

It is suggested that new (as well as potential) users read through the following tutorial. The tutorial uses GULP as the local optimizer to search for stable and metastable TiO_2 structures. The latest source code, installation instructions, and additional tutorials can be found on XtalOpt's website, http://xtalopt.openmolecules.net.

Appendix A.1. Launch XtalOpt

Open Avogadro, go to the "Extensions" menu and select "Crystal Optimization...".

Appendix A.2. Enter composition and restraints

The interface opens to the "Cell Initialization" tab, shown in Figure **??**. We will use a 6 formula unit supercell of TiO_2 for this tutorial, so enter "Ti6 O12" for the cell composition. Let us assume that we know nothing about the system and use very loose restraints. Set all cell length minima to 1 Å and maxima to 20 Å. Constrain the angles to be between 60° and 120°, and the volume from between 1 and 500 $Å^3$. Specify a minimum interatomic distance of 0.5 Å. (Note that due to the angle adjustment described in Section 3.6, 60°-120° is the largest range of cell angles that XtalOpt will generate.)

Appendix A.3. Optimizer setup

On the next tab choose GULP for the local optimizer and enter a template for GULP to use. Select "GULP" as the "Optimization Type" and "GULP .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 samples/gulp-TiO-xtalopt.scheme.

This template implements the potential described in Table 1. It is also included in the download of XtalOpt. 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/Remove" 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. Only one optimization step in this will be used in this tutorial.

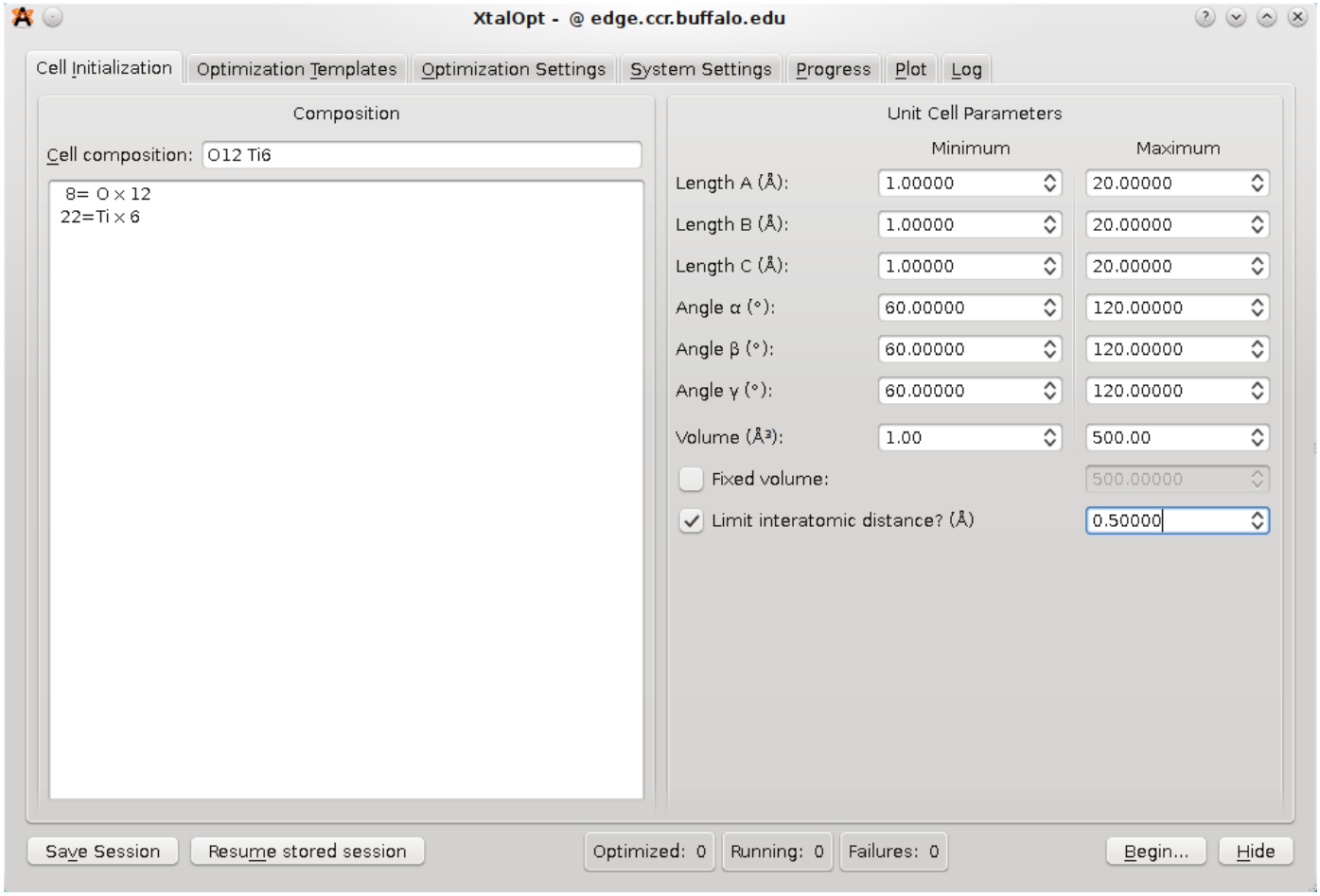


Figure A.6: The "Cell Initialization" tab

Appendix A.4. Optimization settings

In the "Optimization Settings" tab, most of the default settings should suffice (See Table 10). We will be running our test on a dual-core processor, so limit the number of running jobs to 1, and set the number of continuous structures to 2. This is to leave one core open for XtalOpt and the rest of the operating system to run. This is necessary for GULP, which runs locally, but not for VASP or PWSCF, which XtalOpt runs on remote clusters. Initial seeds will not be specified, but the option to do so exists on this screen.

Appendix A.5. System settings

XtalOpt currently will only run GULP on a local installation (although VASP and PWSCF can run on a remote cluster). Thus, for GULP to run we only need to specify the local path and the path to the "gulp" executable. In the "System Settings" tab, we use "/tmp/xtalopt/tutorial" for the path. The path to GULP will depend on your specific installation. This configuration will create a directory for each structure at `/tmp/xtalopt/tutorial/<gen#>x<id#>/` that will contain input, output, and data files specific to each structure. It also uses this information to write two files: `/tmp/xtalopt/tutorial/xtal.qpt`, which contains save/resume information to continue a session that has been stopped, and `/tmp/xtalopt/tutorial/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.

Appendix A.6. "Begin"

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.

Appendix A.7. Monitor progress

As XtalOpt performs the search, the progress table continuously updates, providing information about each structure. In Figure **??** we see individuals in various stages of completion: structure 1x1 has been optimized (in yellow), structure 2x7 has been automatically marked as a duplicate (dark green) of structure 3x3

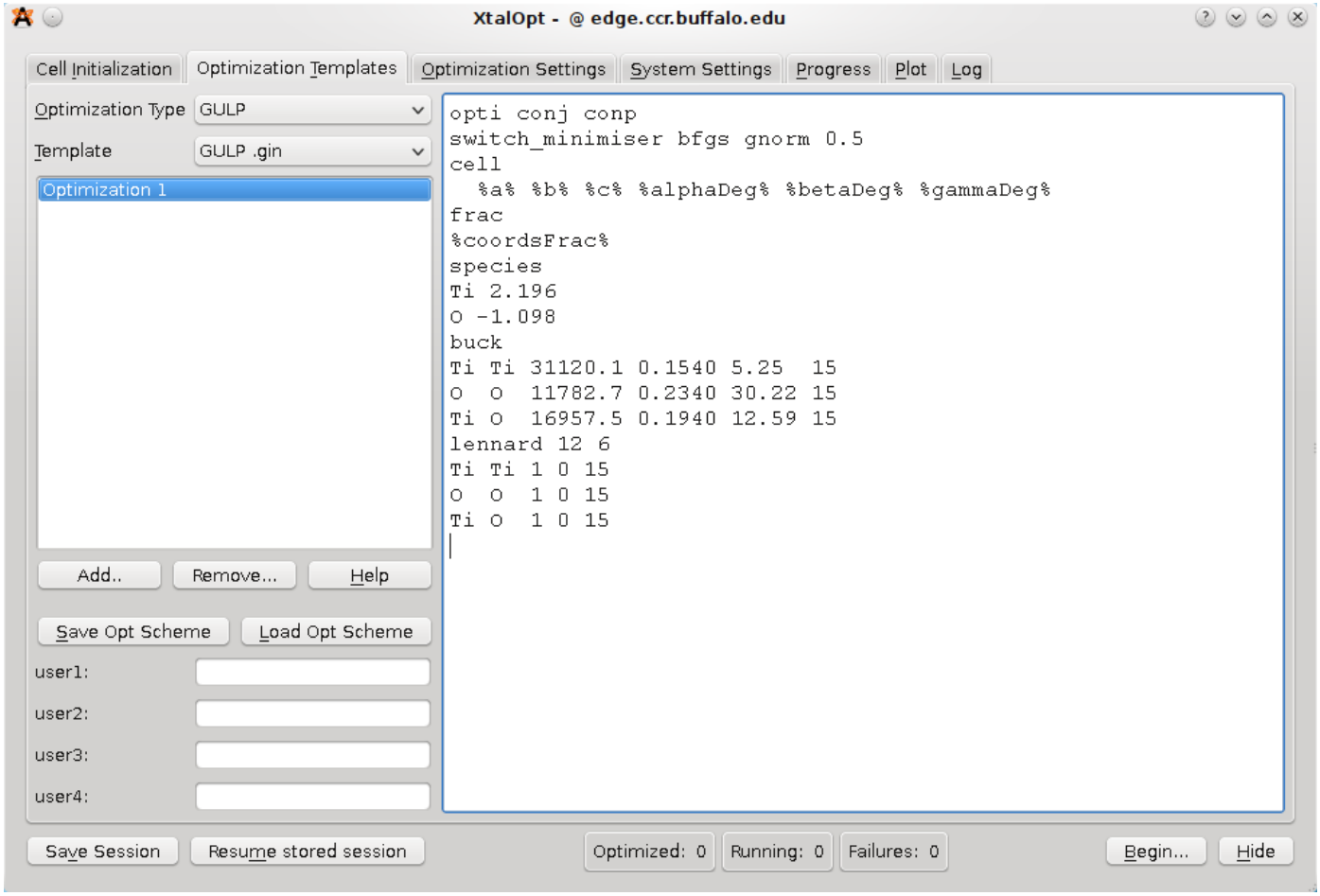


Figure A.7: The "Optimization Templates" tab

and removed from the breeding pool, structure 4x4 is currently undergoing a local optimization (light green), while structure 4x5 is waiting to be optimized (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 ancestory (i.e. parent(s) and parameters for the evolutionary 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.

An additional feature of the progress table is the ability to immediately visualize any of the individuals in the Avogadro main window – simply clicking on a row in this table will display the three-dimensional structure in Avogadro, 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, or replace a problematic structure with a new, random individual.

Appendix A.8. View trends

Another visualization and analysis tool available during the search is the interactive plot. Shown in Figure **??**, 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, as shown in Figure **??**. From this view, one clearly sees 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. From this data set, we see that there is a cluster of very low enthalpy structures around 180 $Å^3$. Armed with this data, we can

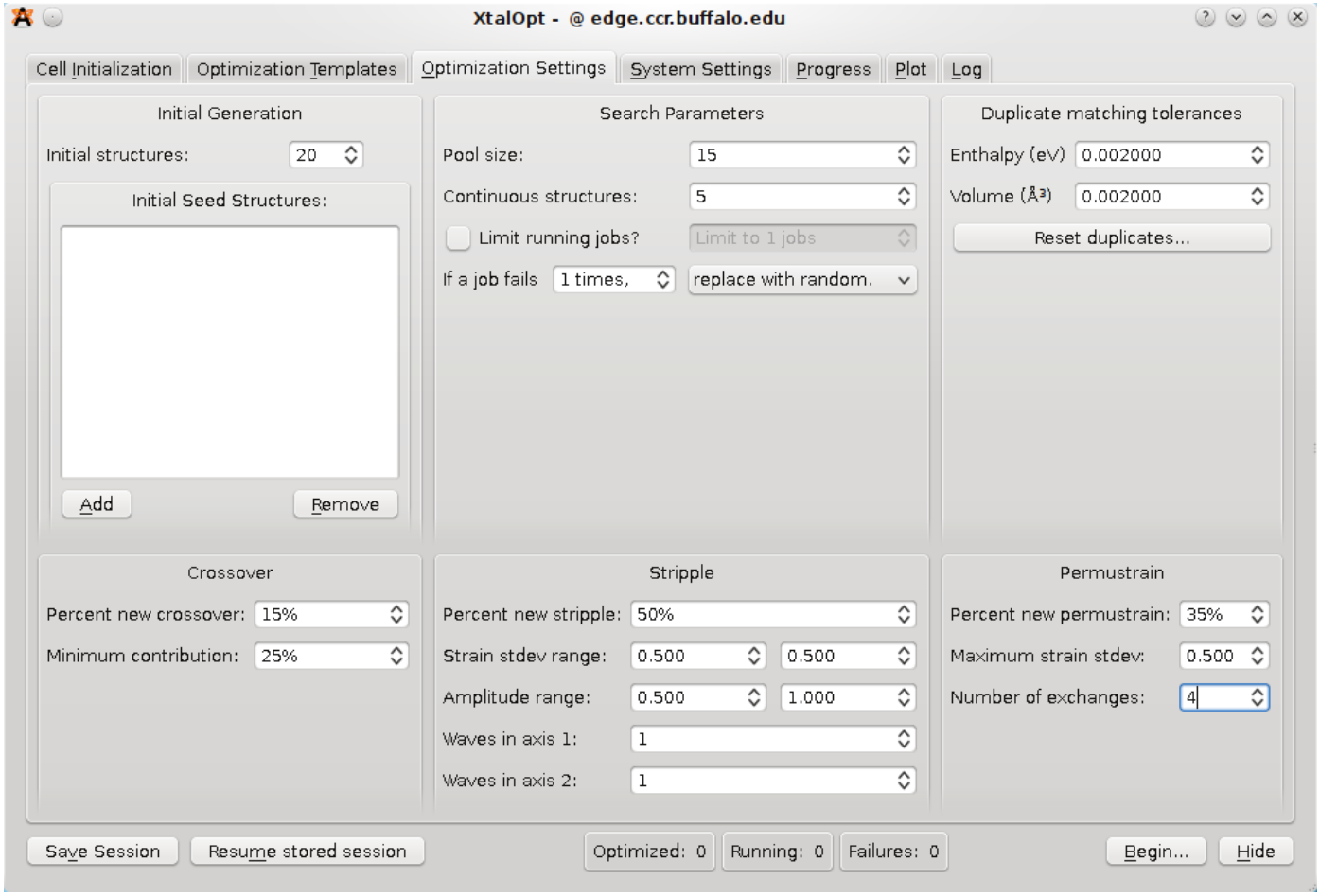


Figure A.8: The "Optimization Settings" tab

update the starting volume on the Cell Initialization tab mid-run to reflect this new piece of information that the search has provided us and enforce this limit on all newly generate structures. 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.

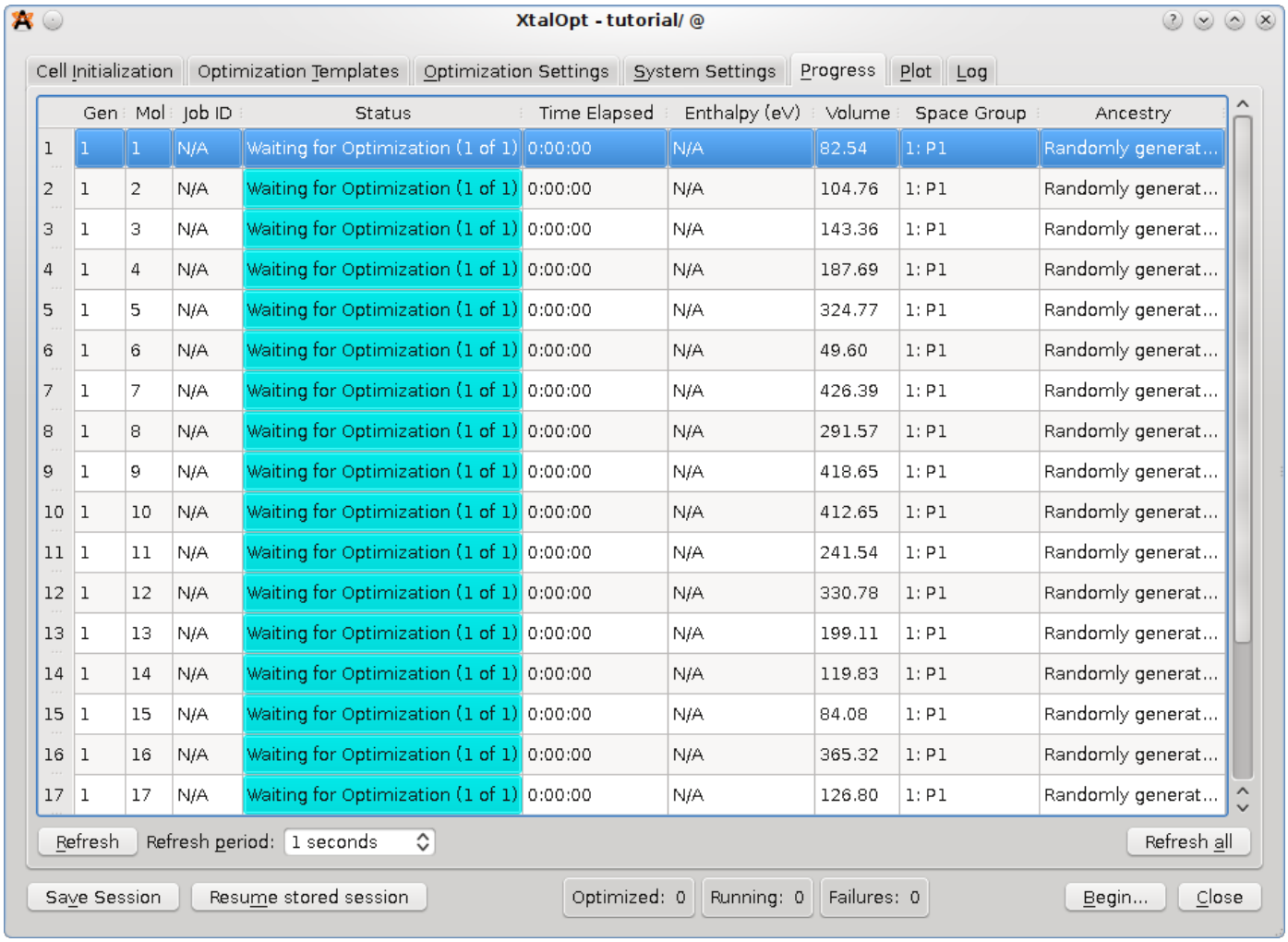
The plot is also interactive; zooming and panning are possible using simple mouse controls. Clicking on a structure's point in the plot will load it in the main Avogadro window, allow all the same functionality as described above in the progress table section.

Appendix A.9. Endpoints

Unfortunately, there are no hard rules for determining when a search should be stopped while investigating an unknown system. Hooper et al.[28] recommend searching until the four best structures are the same for five generations (or approximately 50 structures in the continuous mode). This is no guarantee for success, but provides a starting point. We recommend experimenting with a few known systems to develop a feel for telling when a search completes, and use Hooper's rule as a guideline along the way.

References

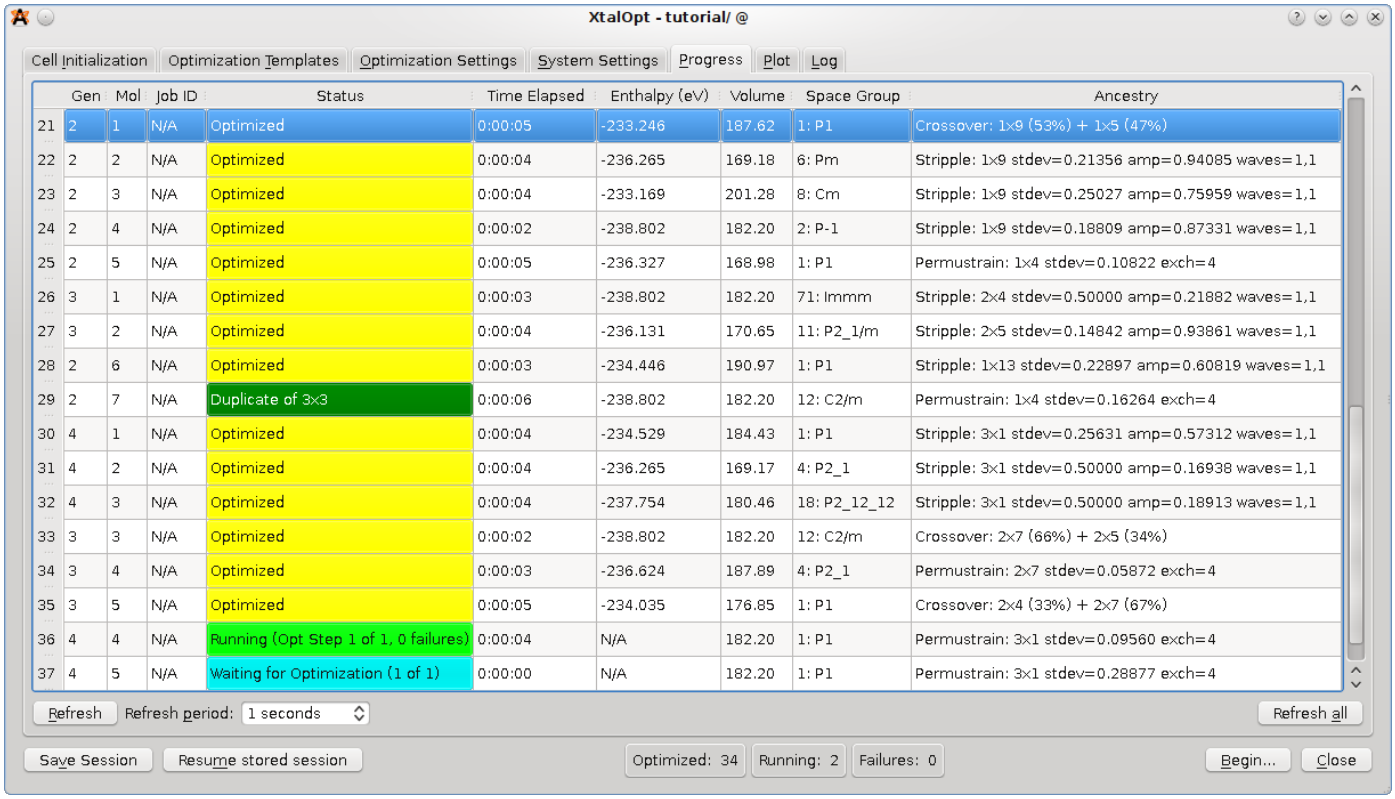
[1] J. Maddox, Nature 335 (1988) 201.
[2] J. Feng, et al., Phys. Rev. Lett. 96 (2006) 017006.
[3] J. Feng, R. G. Hennig, N. W. Ashcroft, R. Hoffmann, Nature 451 (2008) 445.
[4] E. Zurek, R. Hoffmann, N. W. Ashcroft, A. R. Oganov, A. O. Lyakhov, Proc. Nat. Acad. Sci. 42 (2009) 17640.
[5] C. J. Pickard, R. J. Needs, Phys. Status Solidi (B) 246 (2009) 536.
[6] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, Science 220 (1983) 671.
[7] D. J. Wales, J. P. K. Doye, J. Phys. Chem. A 101 (1997) 5111.
[8] P. Raiteri, R. Martonak, M. Parrinello, Angew. Chem. Int. Edit. 44 (2005) 3769.
[9] S. M. Woodley, Phys. Chem. Chem. Phys. 9 (2007) 1070.
[10] W. Paszkowicz, Mater. Manuf. Process. 24 (2009) 174.
[11] B. Hartke, J. Phys. Chem. 97 (1993) 9973.
[12] D. Deaven, K. Ho, Phys. Rev. Lett. 75 (1995) 288.
[13] B. Hartke, J. Comput. Chem. 20 (1999) 1752.
[14] R. L. Johnston, Dalton T. (2003) 4193.
[15] B. Bandow, B. Hartke, J. Phys. Chem. A 110 (2006) 5809.
[16] B. Assadollahzadeh, P. R. Bunker, P. Schwerdtfeger, Chem. Phys. Lett. 451 (2008) 262.
[17] B. Assadollahzadeh, P. Schwerdtfeger, J. Chem. Phys. 131 (2009) 064306.
[18] T. S. Bush, C. R. A. Catlow, P. D. Battle, J. Mater. Chem. 5 (1995) 1269.
[19] S. M. Woodley, P. D. Battle, J. D. Gale, C. R. A. Catlow, Phys. Chem. Chem. Phys. 1 (1999) 2535.
[20] N. Abraham, M. Probert, Phys. Rev. B 73 (2006) 224104.
[21] A. R. Oganov, C. W. Glass, J. Chem. Phys. 124 (2006) 244704.



	Gen	Mol	Job ID	Status	Time Elapsed	Enthalpy (eV)	Volume	Space Group	Ancestry
1	1	1	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	82.54	1: P1	Randomly generat...
2	1	2	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	104.76	1: P1	Randomly generat...
3	1	3	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	143.36	1: P1	Randomly generat...
4	1	4	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	187.69	1: P1	Randomly generat...
5	1	5	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	324.77	1: P1	Randomly generat...
6	1	6	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	49.60	1: P1	Randomly generat...
7	1	7	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	426.39	1: P1	Randomly generat...
8	1	8	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	291.57	1: P1	Randomly generat...
9	1	9	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	418.65	1: P1	Randomly generat...
10	1	10	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	412.65	1: P1	Randomly generat...
11	1	11	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	241.54	1: P1	Randomly generat...
12	1	12	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	330.78	1: P1	Randomly generat...
13	1	13	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	199.11	1: P1	Randomly generat...
14	1	14	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	119.83	1: P1	Randomly generat...
15	1	15	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	84.08	1: P1	Randomly generat...
16	1	16	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	365.32	1: P1	Randomly generat...
17	1	17	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	126.80	1: P1	Randomly generat...

Figure A.9: The “Progress” tab immediately after starting a search

[22] C. Glass, A. Oganov, N. Hansen, Comput. Phys. Commun. 175 (2006) 713.
[23] G. Trimarchi, A. Zunger, Phys. Rev. B 75 (2007) 104113.
[24] G. Trimarchi, A. Zunger, J. Phys. - Condens. Mat. 20 (2008) 295212.
[25] A. R. Oganov, C. W. Glass, J. Phys. - Condens. Mat. 20 (2008) 064210.
[26] N. Abraham, M. Probert, Phys. Rev. B 77 (2008) 134117.
[27] S. Woodley, C. Catlow, Comp. Mater. Sci. 45 (2009) 84.
[28] J. Hooper, A. Hu, F. Zhang, T. Woo, Phys. Rev. B 80 (2009) 104117.
[29] R. Briggs, C. Ciobanu, Phys. Rev. B 75 (2007) 195415.
[30] A. L. S. Chua, N. A. Benedek, L. Chen, M. W. Finnis, A. P. Sutton, Nat. Mater. 9 (2010), 383
[31] Avogadro, development version, http://avogadro.openmolecules.net.
[32] R. Guha, et al., J. Chem. Inf. Model. 46 (2006) 991.
[33] The open babel package, development version, http://openbabel.org.
[34] SPGLIB, development version, http://spglib.sourceforge.net/.
[35] The GNU general public license, http://www.gnu.org/licenses/gpl.html.
[36] Qt - a cross-platform application and UI framework, http://qt.nokia.com/.
[37] G. Kresse, J. Hafner, Phys. Rev. B 47 (1993) 558.
[38] G. Kresse, J. Hafner, Phys. Rev. B 49 (1994) 14251.
[39] G. Kresse, J. Furthmüller, Comp. Mater. Sci. 6 (1996) 15.
[40] G. Kresse, J. Furthmüller, Phys. Rev. B 54 (1996) 11169.
[41] P. Giannozzi, et al., J. Phys. - Condens. Mat. 21 (2009) 395502.
[42] J. D. Gale, Philos. Mag. B 73 (1996) 3.
[43] J. D. Gale, A. L. Rohl, Mol. Simulat. 29 (2003) 291.
[44] J. D. Gale, J. Chem. Soc., Faraday T. 93 (1997) 629.
[45] N. Benedek, A. Chua, C. Elsässer, A. Sutton, M. Finnis, Phys. Rev. B 78 (2008) 064110.
[46] M. I. McMahon, R. J. Nelmes, Chem. Soc. Rev. 35 (2006) 943.
[47] M. I. McMahon, R. J. Nelmes, S. Rekhi, Phys. Rev. Lett. 87 (2001) 255502.
[48] R. J. Nelmes, M. I. McMahon, J. S. Loveday, S. Rekhi, Phys. Rev. Lett. 88 (2002) 155503.
[49] O. Degtyareva, M. I. McMahon, D. R. Allan, R. J. Nelmes, Phys. Rev. Lett. 93 (2004) 205502.
[50] Z. H. Li, A. W. Jasper, D. G. Truhlar, J. Am. Chem. Soc. 129 (2007) 14899.
[51] A. R. Oganov, M. Valle, J. Chem. Phys. 130 (2009) 104504.
[52] R. D. Shannon, C. T. Prewitt, Acta. Cryst. B25 (1969) 925.
[53] E. Jones, SciPy: Open source scientific tools for Python, 2001.
[54] J. Zhang, L. Zhang, T. Cui, Y. Li, Z. He, Y. Ma, G. Zou, Phys. Rev. B 75 (2007) 104115.
[55] S. J. Duclos, Y. K. Vohra, A. L. Ruoff, S. Filipek, B. Baranowski Phys. Rev. B 36 (1987) 7664.



	Gen	Mol	Job ID	Status	Time Elapsed	Enthalpy (eV)	Volume	Space Group	Ancestry
21	2	1	N/A	Optimized	0:00:05	-233.246	187.62	1: P1	Crossover: 1x9 (53%) + 1x5 (47%)
22	2	2	N/A	Optimized	0:00:04	-236.265	169.18	6: Pm	Stripple: 1x9 stdev=0.21356 amp=0.94085 waves=1,1
23	2	3	N/A	Optimized	0:00:04	-233.169	201.28	8: Cm	Stripple: 1x9 stdev=0.25027 amp=0.75959 waves=1,1
24	2	4	N/A	Optimized	0:00:02	-238.802	182.20	2: P-1	Stripple: 1x9 stdev=0.18809 amp=0.87331 waves=1,1
25	2	5	N/A	Optimized	0:00:05	-236.327	168.98	1: P1	Permustrain: 1x4 stdev=0.10822 exch=4
26	3	1	N/A	Optimized	0:00:03	-238.802	182.20	71: Immm	Stripple: 2x4 stdev=0.50000 amp=0.21882 waves=1,1
27	3	2	N/A	Optimized	0:00:04	-236.131	170.65	11: P2_1/m	Stripple: 2x5 stdev=0.14842 amp=0.93861 waves=1,1
28	2	6	N/A	Optimized	0:00:03	-234.446	190.97	1: P1	Stripple: 1x13 stdev=0.22897 amp=0.60819 waves=1,1
29	2	7	N/A	Duplicate of 3x3	0:00:06	-238.802	182.20	12: C2/m	Permustrain: 1x4 stdev=0.16264 exch=4
30	4	1	N/A	Optimized	0:00:04	-234.529	184.43	1: P1	Stripple: 3x1 stdev=0.25631 amp=0.57312 waves=1,1
31	4	2	N/A	Optimized	0:00:04	-236.265	169.17	4: P2_1	Stripple: 3x1 stdev=0.50000 amp=0.16938 waves=1,1
32	4	3	N/A	Optimized	0:00:04	-237.754	180.46	18: P2_12_12	Stripple: 3x1 stdev=0.50000 amp=0.18913 waves=1,1
33	3	3	N/A	Optimized	0:00:02	-238.802	182.20	12: C2/m	Crossover: 2x7 (66%) + 2x5 (34%)
34	3	4	N/A	Optimized	0:00:03	-236.624	187.89	4: P2_1	Permustrain: 2x7 stdev=0.05872 exch=4
35	3	5	N/A	Optimized	0:00:05	-234.035	176.85	1: P1	Crossover: 2x4 (33%) + 2x7 (67%)
36	4	4	N/A	Running (Opt Step 1 of 1, 0 failures)	0:00:04	N/A	182.20	1: P1	Permustrain: 3x1 stdev=0.09560 exch=4
37	4	5	N/A	Waiting for Optimization (1 of 1)	0:00:00	N/A	182.20	1: P1	Permustrain: 3x1 stdev=0.28877 exch=4

Figure A.10: The “Progress” tab mid-run

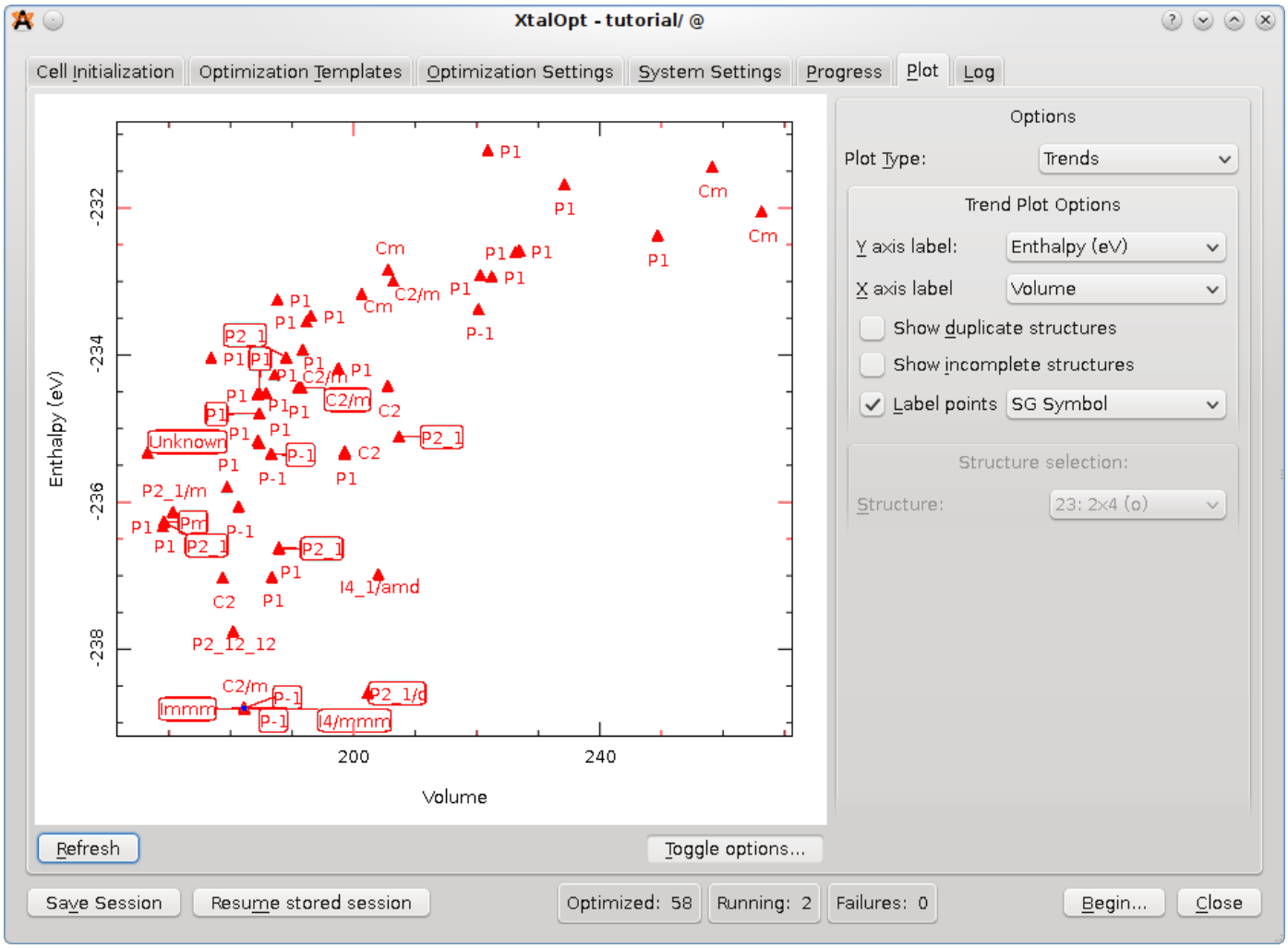


Figure A.11: The “Plot” tab mid-run displaying enthalpy vs. volume. Each structure is labeled with its Hermann Mauguin spacegroup symbol.