A NEW FRAMEWORK FOR MATERIAL INFORMATICS AND ITS APPLICATION TOWARD ELECTRIDE-HALIDE MATERIAL SYSTEMS

Jack D. Sundberg

A dissertation submitted to the faculty at the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Chemistry.

Chapel Hill
2022

ABSTRACT

Jack D. Sundberg: A new framework for material informatics and its
application toward electride-halide material systems
(Under the direction of Scott C. Warren)

Despite many years of material exploration, the vast majority of unique crystalline materials remain undiscovered, and these undiscovered materials may offer stronger steels, better catalysts, improved transistors, and many other solutions to urgent societal problems. We therefore need a fast and efficient way of identifying new materials so that society can harness their benefits. To aid in accelerated materials discovery, this dissertation describes a computational framework designed for high-throughput calculations and analyses: the Simulated Materials Ecosystem (Simmate). This software allows users to explore various crystal databases, predict new materials, quickly calculate properties, and share results across analyses. We illustrate Simmate's functionality through the exploration of an exotic class of materials known as electrides, which have gained considerable attention in recent literature thanks to their applications as superconductors, co-catalysts, and solid-state dopants. This diverse set of applications derives from an electride's defining property: bare electrons that exist at isolated lattice sites. "Electride electrons" effectively serve as anions, which led us to propose the direct substitution of electrides with other -1 species, namely, halides (F^-, Cl^-, Br^-, I^-). Herein, we use Simmate to explore electride-halide systems, understand transitions between such materials, and predict new systems with enhanced material properties. This work ultimately led to the identification of novel ionic conductors, metastable electrides, and new search algorithms for discovering more of the same. Our framework and high-throughput search strategies are highly generalizable and will accelerate the exploration of many different materials beyond our illustrative examples with electride-halide material systems.

To my family and friends, I could not have done this without your love and support.

ACKNOWLEDGMENTS

My mentors, colleagues, family, and friends – all of you have done so much to support me in my work and have always been my source of inspiration. Thank you for being at my side during my graduate school journey. I am truly lucky to have you all in my life.

First, I would like to acknowledge those that I worked closest with. There was a large and fun group of characters during my time in the Warren lab: Kaci, Tyler, Adam, Zack, Dan, Jacob, Madeline, Sean, Lauren, Jessica, Matt, Becca, Connor, Duncan, Karen, Don, and Siona. Each of you made the lab an exciting place to be. One of the things I enjoyed most was watching our off-the-cuff discussions grow into full projects, especially after years of refining an idea. And, of course, I will always look back fondly on the times we gave each other much needed distractions. I'm glad to share all those moments – from our little games in lab to our long chats on hobbies and interests. Don't ever stop goofing around and always keep spirits high. Your research and success will surely follow.

To my advisor, Scott, thank you for being such an amazing mentor over the years and for teaching me the importance of passion in science. Having the freedom to chase down ideas and explore my interests ultimately led to my most exciting work, and I am certain that I would not have made such an impact without your support. I will forever be grateful for your mentorship. It has been amazing to watch the lab grow and tackle so many new challenges, and I can't wait to see the next set of breakthroughs.

I would also like to thank Professors Jim Cahoon, Yosuke Kanai, Jillian Dempsey, Megan Jackson, Jeffery Dick, and Jerry Meyer for serving on my preliminary and final defense committees. For some, our overlap may have been only over a few meetings, but I can assure you that your guidance always showed through your own students, whom I have grown close to over the years. I am extremely fortunate to have you all review my research from such diverse and knowledgeable backgrounds.

Then, there are my partners-in-crime, my friends. From every major stage in my life – Fairview, Denison, and UNC – each of you has given me continued support in my work, and it has meant the world to me. I surely would not have made it through graduate school without our spontaneous trips and adventures. In particular, I would like to thank those part of my UNC chemistry program that I became closest with. Staying in for crafts, cocktails, and boardgames will always be my favorite nights while in Chapel Hill. It’s been a blast, and I look forward to seeing where the next steps take us.

Finally, thank you to my amazing family. Mom and Dad, you have always been my shoulders to lean on, guiding me through all major milestones in my life. Everything I have worked so hard to accomplish starts with you. Thank you. Michael and Jessica, having you close by has meant the world. Thank you so much for the life-saving meals and weekend escapes. Of all things in North Carolina, I will miss the quick drive down to Lulu’s (+ Finn’s) palace the most. Kendee and Buttercup, you both will always be my first call for a quick laugh and chat when I need it. Always keep chasing your passions as it inspires me to do the same.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND SYMBOLS

ASE	Atomic Simulation Environment
AFLOW	Automatic-FLOW for Materials Discovery
API	application programming interface
BS	band structure
CI	continuous integration
CIF	crystallographic information file
CLI	command line interface
COD	The Crystallography Open Database
COOP	crystal orbital overlap populations
DAG	directed acyclic graph
DDI	decoupled, dynamic, and iterative
DFT	density functional theory
DOS	density of states
E_{approx}	energy barrier calculated using several approximations
E_{hull}	hull energy
E_{NEB}	energy barrier calculated by nudged elastic band
EAX	electride-anion exchange
EHAP	electride-halide analog pair
EHMS	electride-halide material system
ELF	electron localization function
EXT	external integration (or function wrapping)
F_{rel}	relative magnitude of the atomic forces for a crystal
FIB	fluoride-ion battery
ΔG_{rxn}	Gibbs free energy of reaction
ΔG_{solv}	Gibbs free energy of solvation
GGA	generalized gradient approximation
GraphQL	graph query language

HPC	high performance computing (cluster)
IDPP	image-dependent pair potential
IO	input-output
ΔIRO	change in ionic radii overlap
JARVIS	Joint Automated Repository for Various Integrated Simulations
JSON	JavaScript object notation
LIB	lithium-ion battery
MD	molecular dynamics
NEB	nudged elastic band
NOMAD	The Novel Materials Discovery Laboratory
OQMD	The Open Quantum Materials Database
PAW	Projector-augmented wave
PBE	Perdew–Burke-Ernzerhof
PC	personal computer
PDOS	projected density of states
pRDF	partial radial distribution function
PyMatGen	Python Package for Material Genomics
RDF	radial distribution function
REST	representational state transfer
RT	room temperature
S3Workflow	supervised-staged-shell workflow
Simmate	The Simulated Materials Ecosystem
SQL	structured query language
SSH	secure shell protocol
VASP	Vienna ab initio software package

CHAPTER 1: INTRODUCTION

1.1 Motivation

Throughout history, new materials have spurred the realization of improved technologies and led to massive societal changes. Our future will be no exception. Yet-undiscovered materials will be essential to address urgent social problems such as climate change and sustainable energy use.[1] Whether we target materials to create new technologies, to improve upon existing architectures, or to advance chemical understanding, our exploration of materials must be as fast and efficient as possible.

However, materials discovery remains a slow and expensive process. The vast majority of materials (likely >99.9%) remain unexplored, and the time required for researchers to fully understand a single material can span years or decades.[2,3] It can take even longer to identify ideal, high-performance materials for a technology and optimize such systems.[4] This is because materials discovery is greatly complicated by multidimensional parameter space – both in the number of possible compositions and the complexity of phase competition.[2,5]

Detailed exploration of all possible materials is not practical, so materials discovery must be guided by chemical intuition and computational predictions. Researchers should utilize all data available to make informed choices on which materials are most promising. Historically, however, data-driven decisions were not possible due the small amount of preexisting data. Now, because of advancements in hardware and software, materials discovery has begun shifting into the big data era.[6,7,8,9] Several large-scale initiatives push for the generation of materials data, and these have led to massive databases with >1,000,000 crystalline materials and their properties.[10,11,12,13,14,15,16,17] Now more than ever, researchers can rely on data-informed decisions in their everyday research.

Moreover, materials science data is not limited to well-developed fields or experimentally known materials. In fact, high-quality data can be quickly generated for systems that have yet to be synthesized.

This provides an important approach to accelerate the arrival of new materials, technologies, and understanding.

Alternative battery systems are an ideal example of this. Decades of research have led to lithium-ion batteries (LIBs) becoming transformative to our everyday lives, but there is nevertheless an urgency to identify alternative setups to meet growing energy demands or resource limits.[18,19.20] This includes other cation-shuttle batteries (Na^+, Mg^{2+}, etc.) or, more recently, anion-shuttle batteries (F^- or Cl^-).[21,22] For the latter, the first reversible fluoride-ion battery (FIB) was reported only in 2011, and although they have the potential to outperform LIBs, FIB technology has not yet matured.[23,24] There is still much to understand before FIBs reach commercial use – such as fluoride transport, chemical stabilities, or even just ideal material systems. Although little data exists for FIBs, computational methods enable us to make informed decisions while optimizing the technology. Computation allows us to explore novel chemical compositions, predict thermodynamic/kinetic stabilities, and screen material properties before any experimental work is done – and at a significantly faster rate than is possible with synthesis.

Together, the necessities of data-informed science and of developing new technologies serve as the motivations of this dissertation. This work seeks to accelerate broad materials exploration and understanding, so many of the methods introduced are generalizable beyond FIBs. To demonstrate the concepts and methods, I explore novel halide systems and their closely related counterparts, where I show how property predictions are effective in exotic systems with limited experimental data. Through the generation of new data and the mining of existing information, I illustrate how materials discovery leverages informatics and data science to accelerate the realization of new technologies and understanding.

In this chapter, I introduce the four key subjects of this dissertation: material informatics, electride-halide material systems, transport in FIBs, and evolutionary structure prediction. Further, I briefly discuss the broader perspective/process of materials discovery that motivates the mixture of method development and applied science within this dissertation. Thus, this chapter provides the context and components that propel the work in the following chapters.

1.2 Material Informatics

When imagining the intersection of data science and materials chemistry, one typically thinks of machine learning, artificial intelligence, or deep learning. However, to realize these applications, there must first be large amounts of clean, accessible materials data.[25] Material informatics serves as the connection between materials science and the many exciting ways data can be applied. Specifically, material informatics describes the creation (or collection), organization, and distribution of materials science data.

Materials science data can be anything from experimental spectra to ab initio calculations, and the diversity of information has led to many separate collection and standardization efforts. Some initiatives focus on specialized cases such as XPS spectra, while other projects are dedicated to fully theoretical calculations of prototype structures.[26,27,28,29] However, the introduction of many smaller, specific standardizations introduces a major problem: it becomes challenging to navigate multiple datasets because one must learn and adjust to each format first.

Therefore, the largest and most successful initiatives seek to capture broad data types under more encompassing definitions.[10,30] This lowers the barrier for entry to others and facilitates the incorporation of larger datasets.[31] Inclusion ultimately leads to databases that include both experimental and computational data, as well as data from many different sources (different ab initio software or diverse characterization techniques). Data organization therefore needs to account for many diverse scenarios and constantly evolve to account for new and unexpected cases.

Scrutinizing data organization might seem overdone in some scenarios, but it ultimately enables more effective collection and distribution of data. For example, by generalizing the ab initio results of a type of calculation (e.g., relaxations, dynamics, electronic structure, etc.), one can leverage results from different codes (e.g., VASP, Quantum Espresso, ABINIT, etc.) into a single database.[15,30] Furthermore, a unified database simplifies the distribution of data through application or website interfaces. When data collection, organization, and distribution are handled within a single codebase, one can quickly build out complex features and analyses that would otherwise not be possible.[32]

To this end, this dissertation introduces a new framework for material informatics. With materials discovery as our primary objective, we build out a suite of functions that handle common calculations, data storage, and user interfaces, where each component prioritizes the ability to scale for millions of data entries. We illustrate the utility of material informatics in the context of several example material systems as well as in the development of new highly parallel algorithms.

1.3 Electride-Halide Material Systems

Electrides have gained considerable attention in recent literature thanks to their applications as superconductors, cocatalysts, and solid-state dopants. To this end, a single electride material (Ca_2N) has demonstrated conductivities competitive with silver[33], catalytic activity for ammonia production at low temperature[34], and sufficient electron doping to induce a phase change in $MoTe_2$.[35] These applications derive from an electride's defining property: bare electrons that exist at isolated lattice sites. These lone electrons are often referred to as anionic electrons, making a compound such as Ca_2N more accurately represented with the formula $[Ca_2N]^+ \cdot e^-$ (where the oxidation states of Ca and N are 2^+ and 3^-, respectively).

This atypical motif is more easily understood by comparing an electride to its halogenated analog. For example, $[Ca_2N]^+ \cdot e^-$ possesses the chloride analog Ca_2NCl. In this electride-halide analog pair (EHAP), the structures can be viewed as substitution derivatives of e^- for Cl^- and vice versa. Here, the EHAP is that of a chloride analog, whereas the full electride-halide material system (EHMS) is composed of all halide analogs for a given electride (e.g., Ca_2N with Ca_2NX; X = F, Cl, Br, I) (Figure 1-1). Previously in literature, an EHAP has strictly been used to aid in describing an electride; however, in this dissertation, the idea of an EHMS is introduced and developed towards (i) synthesizing novel electrides and (ii) producing halide-ion batteries.

Electrides have been exclusively made via high-temperature bulk synthesis and within systems where the target electride is the thermodynamically favored phase.[36,37] Even in these cases, the majority of electrides represent extremely challenging syntheses due to competing phases.[38] Inspection of the EHMS suggests an alternative synthetic approach: low-temperature dehalogenation of the halide analog of an electride. If the halide is mobile, one could drive the halide out of the material and form the electride analog as a result. Whether this is done via electrochemical or alkali-vapor etching, the low-temperature synthesis avoids decomposition to competing phases thanks to kinetic trapping.

The low-temperature shuttling of halides within EHMS could therefore lead to the synthesis of previously inaccessible electrides. This approach is especially appealing for metastable electrides where high-temperature bulk synthesis would result in the decomposition of the desired product and a different phase altogether. One of the objectives of this dissertation is therefore to propose a series of candidate halide materials for low-temperature deintercalation.

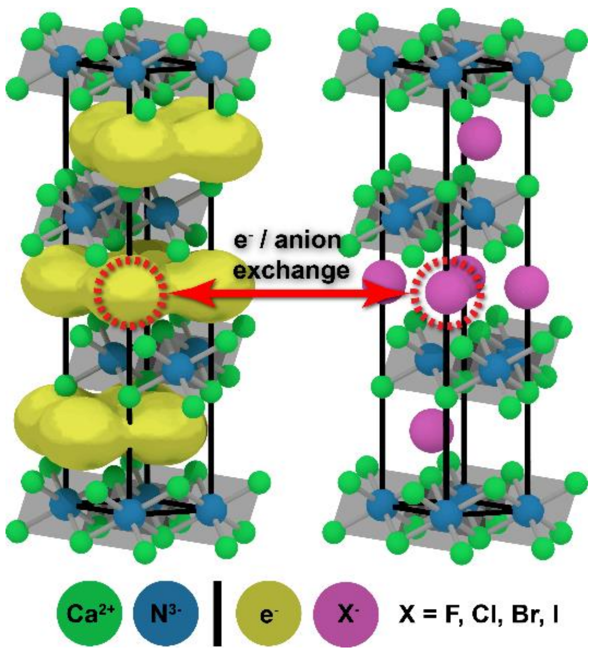


Figure 1-1. An example electride-halide material system. $[Ca_2N]^+\cdot e^-$ (left) with an arbitrary halide analog Ca_2NX (right). The pair represents an EHAP, as they are isostructural – only differing by substitution of e^- for X^-. All anion derivatives (e^-, F^-, Cl^-, Br^-, I^-) together make up the EHMS.

1.4 Fluoride-Ion Batteries and EAX

Halide batteries are a new and underdeveloped technology relative to cation-shuttle batteries (e.g., LIBs).[21,22,24] However, they have mostly relied on the same fundamental electrochemical principles. Lithium (Li^+) has received the most attention because it is the smallest cation, and likewise, most literature on anion-shuttle batteries is focused almost exclusively on the smallest halide, fluoride (F^-).[22] The first fluoride ion battery (FIB) was reported in 2011, but it required an operating temperature of 150 °C because the solid electrolyte displayed poor conductivity at room temperature.[23]

Since this pioneering study, there has been a push for improved electrodes and electrolytes, but the search has been almost exclusively on alkaline-earth fluorides (fluorite-type crystals), rare-earth metal fluorides (tysonite-type crystals), and closely related systems.[24,39,40] These structure types exhibit poor intrinsic conductivities as pure phases and still often require elevated operating temperatures. Empirical optimization via aliovalent doping, ball milling, and sintering has been effective in producing a few solid electrolytes suitable for room-temperature FIBs, but a better understanding of fluoride transport is still needed.[41,42,43,44,45,46]

Furthermore, known FIB materials still suffer from poor electrochemical stabilities and chemical compatibilities. Conversion-based metal fluoride electrodes (e.g., CuF_2 and BiF_3 cathodes; CeF_3, CaF_2, and MgF_2 anodes) remain dominant in existing studies due to their high theoretical capacities, despite their poor cyclability caused by large volume changes throughout charging/discharging.[24] Alternatively, intercalation-based electrodes offer improved cyclability but are currently limited to four structure types: Ruddlesden–Popper (K_2NiF_4-type)[47], Schafarzikite (MSb_2O_4)[48], anion-deficient perovskite ($AMO_{3}-\gamma$)[49], and layered rocksalts (MoS_2-type)[50]. These systems each face critical limitations, such as irreversibility for Schafarzikite structures or even lack of chemically compatible solid electrolytes for layered rocksalts.

More recently, we have predicted that EHMSs possess unique characteristics that make them ideal for reversible halide-ion batteries.[50] Specifically, we found that electride character greatly affects the diffusion mechanism and, in many cases, leads to highly mobile fluoride. The presence of electride electrons helps to stabilize the transition state during ionic transport, and further, electride-anion exchange (EAX) occurs via a substitution-like process (Figure 1-2). This means that the host lattice does not undergo redox during (de)intercalation, which contrasts with all previously known cation and anion-shuttle battery materials. EHAPs such as Y2C-Y2CF_2 show great promise as high-voltage, high-capacity electrodes for FIBs due to EAX, and the discovery of similar compounds will help release high-performance, room-temperature FIBs.[50, 51]

The discovery of EAX also highlights the need for a better understanding of halide diffusion. It remains unclear whether EAX is limited to materials with strong electride character or if EAX-like mechanisms can be realized in a range of halide systems. It is likely that there are many undiscovered materials that demonstrate high fluoride mobility due to this mechanism. We therefore explore halide systems (specifically fluorides) to identify further instances of EAX as well as identify new, robust materials for FIBs.

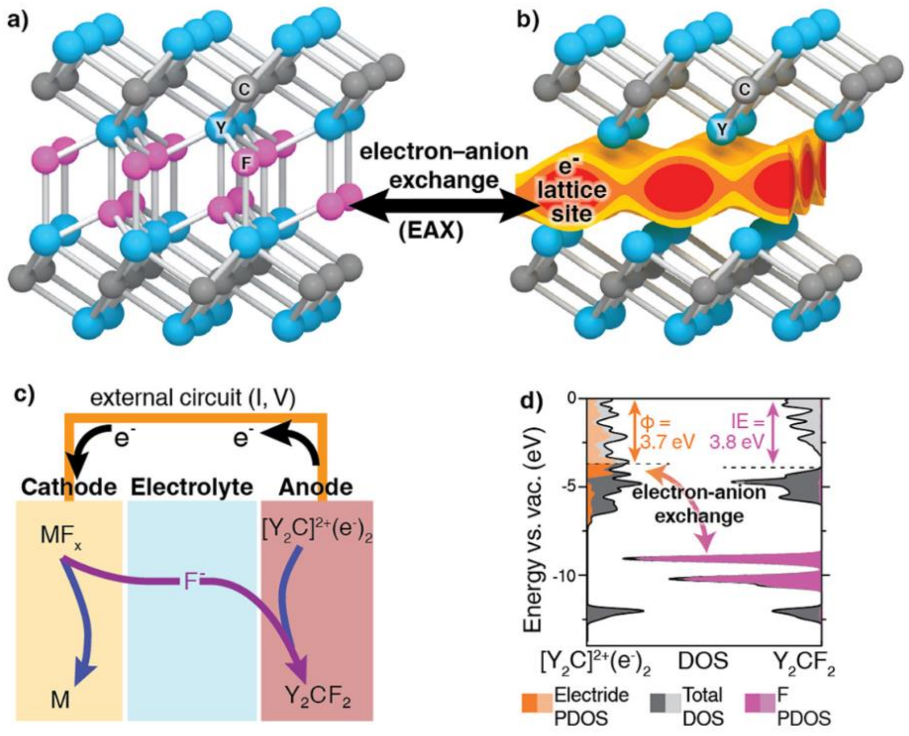


Figure 1-2. Concept of electron-anion exchange (EAX). The reversible conversion of (a) Y_2CF_2 to (b) $[Y_2C]^{2+}(e^-)_2$ under an applied bias, where yellow, orange, and red show isosurfaces of the anionic electrons at the Fermi level. (c) Battery configuration incorporating $[Y_2C]^{2+}(e^-)_2$ as an anode, a F^- electrolyte, and a cathode (e.g., a metal fluoride). (d) Effect of electron-anion exchange on the partial density of states in the Y_2C system, where IE is the ionization energy and Φ is the work function.

1.5 Evolutionary Structure Prediction

One can quickly propose many possible EHMSs to explore, which progress to hundreds or thousands of candidate materials for EAX or FIBs. However, the majority of these materials may not be thermodynamically stable or experimentally synthesizable. In fact, it is exceedingly rare for materials of an EHMS (the electride plus all halide analogs) to all be experimentally known, based on existing material databases.[11] It is therefore essential to have a rapid process to prioritize easily synthesizable phases/systems. Rather than going through candidate materials via experimental trial and error, it is far more efficient to computationally predict the thermodynamic stability of promising materials.

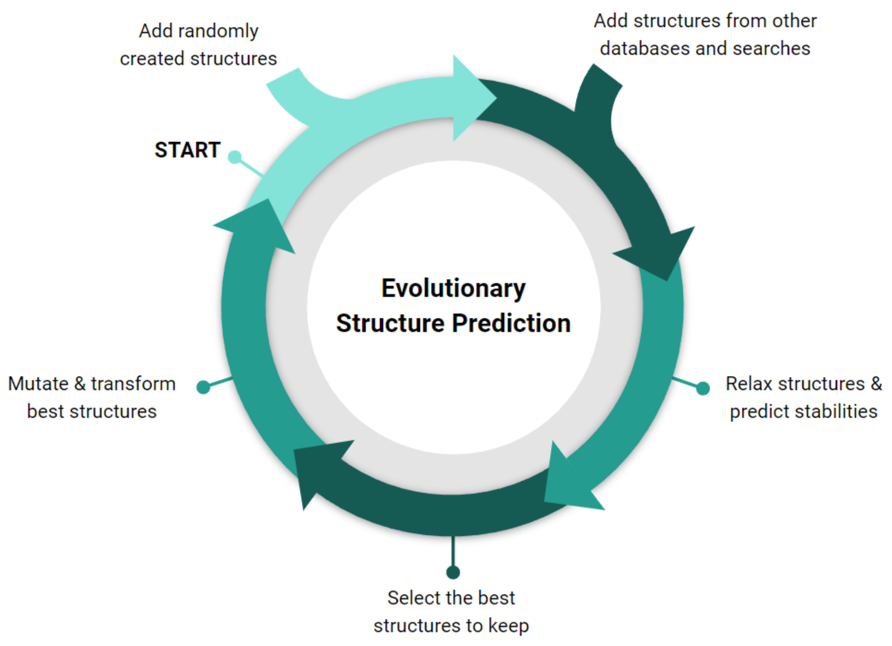

Add randomly created structures
Add structures from other databases and searches
START
Evolutionary Structure Prediction
Mutate & transform best structures
Relax structures & predict stabilities
Select the best structures to keep

Figure 1-3. Illustration of the evolutionary structure prediction cycle. The cycle begins with randomly created structures and (optionally) adds structures from other crystal databases or even other similar evolutionary searches. Ab initio calculations are then used to locally optimize each candidate structure and evaluate its stability/energy. The best structures are then used to generate new best-guess candidates that are evaluated in the next cycle. This process continues until the ground state structure is found.

There are strategies to search for the lowest energy structure for a composition, such as using prototype libraries[27,28,29], meta-dynamics[52,53] or even Monte Carlo sampling[54]. Of the many approaches, ground state phases are most commonly predicted using evolutionary search algorithms.[55] Here, phase space is explored through a cycle of (i) randomly creating structures, (ii) locally optimizing structures to determine their energy, and (iii) using the best structures to generate a new ‘improved’ set of structures. This cycle continues until the ground state structure is found (Figure 1-3), and the overall search can require thousands of ab initio calculations and millions of CPU hours. As a result, existing evolutionary codes struggle to explore beyond simple systems (e.g., a single-phase diagram containing two elements) due to computational cost.

Thanks to advancements in material informatics, there is now an opportunity to accelerate evolutionary structure prediction using prototype libraries, material databases, crystal toolkits, modern workflow engines, and more. These advances have largely remained independent from existing evolutionary materials discovery software.[56,57,58,59,60] Initial attempts to improve data processing in evolutionary searches have already shown that systems can be explored orders of magnitude faster, but this was limited to file-based transfer of information between legacy code runs.[61,62] There is currently no existing evolutionary package that fully utilizes modern databases, APIs, and frameworks. However, by integrating these features, evolutionary structure prediction can leverage (a) automatic integration of third-party data and software, (b) data sharing across separate evolutionary searches, and (c) asynchronous scaling and distribution of calculations across arbitrary resources.

These improvements can have a profound impact on how solid-state chemistry is carried out for novel chemical systems. On HPC clusters that parallelize hundreds of multicore calculations, predicting the thermodynamic stability of phase space can take a week for a single composition and over a month for a binary chemical system. However, by reducing this time to days or even under an hour, researchers can efficiently identify promising chemical systems/phases and add confidence to their thermodynamically driven syntheses. This dissertation therefore introduces a new evolutionary structure prediction software that integrates modern practices in material informatics. This ultimately leads to the efficient exploration of EHMSs that advance our understanding of FIBs and EAX.

1.6 Outlook

The application of materials informatics does not end with EHMSs, FIBs, and evolutionary algorithms. These are simply exemplary scenarios where we have already applied our framework. Generalized workflows, databases, and user interfaces can serve to accelerate materials discovery in many other contexts. This is because materials discovery relies on a constantly evolving cycle of phase space exploration, property predictions, and experimental characterization. Many components can be derived from the results (new datasets, theoretical models, or promising materials), and new methods can be added to accelerate the process.

There are far-reaching implications when data is utilized properly. We illustrate this by discussing how search strategies (even those with drastically different objective functions) can collaborate and share results. While it is outside the scope of this text, our lab is extending the framework to many new and exciting applications, such as machine-learned potentials, modeling disorder, cluster expansion, and even fitting of experimental spectra. Therefore, throughout this dissertation, keep in mind the importance of generalization and the potential applications for materials discovery that go beyond our example systems.

REFERENCES

(1) Semieniuk, G., Taylor, L., Rezai, A. et al. Plausible energy demand patterns in a growing global economy with climate policy. *Nat. Clim. Chang.* **11**, 313–318 (2021).

(2) DiSalvo, F. J. Solid-State Chemistry: A Rediscovered Chemical Frontier. *Science* **247**, 4943, 649–655. (1990).

(3) Sun, W. et al. The thermodynamic scale of inorganic crystalline metastability. *Sci. Adv.* **2**, 11, e1600225 (2016).

(4) Alberi, K., et al. The 2018 materials by design roadmap. *J. Phys. D: Appl. Phys.* **52**, 013001 (2018).

(5) Artem R. Oganov and Colin W. Glass, Crystal structure prediction using ab initio evolutionary techniques: Principles and applications. *J. Chem. Phys.* **124**, 244704 (2006).

(6) Pyzer-Knapp, E.O., Pitera, J.W., Staar, P.W.J. et al. Accelerating materials discovery using artificial intelligence, high performance computing and robotics. *npj Comput Mater* **8**, 84 (2022).

(7) Suh, C., Fare, C., Warren, J. A. & Pyzer-Knapp, E. O. Evolving the materials genome: how machine learning is fueling the next generation of materials discovery. *Annu. Rev. Mater. Res.* **50**, 1–25 (2020).

(8) Tabor, D. P. et al. Accelerating the discovery of materials for clean energy in the era of smart automation. *Nat. Rev. Mater.* **3**, 5–20 (2018).

(9) Lu, Z. Computational discovery of energy materials in the era of big data and machine learning: A critical review. *MRE* **1**, 3, 100047 (2021).

(10) de Pablo, J.J., Jackson, N.E., Webb, M.A. et al. New frontiers for the materials genome initiative. *npj Comput Mater* **5**, 41 (2019).

(11) Jain, A. et al. Commentary: The Materials Project: A materials genome approach to accelerating materials innovation. *APL Mater*. **1**, 011002 (2013).

(12) Curtarolo, S. et al. AFLOW: an automatic framework for high-throughput materials discovery. *Comp. Mater. Sci.* **58**, 218–226 (2012).

(13) Saal, J. E., Kirklin, S., Aykol, M., Meredig, B. & Wolverton, C. Materials design and discovery with high-throughput density functional theory: the open quantum materials database (OQMD). *JOM* **65**, 1501–1509 (2013).

(14) Choudhary, K., Garrity, K.F., Reid, A.C.E. et al. The joint automated repository for various integrated simulations (JARVIS) for data-driven materials design. *npj Comput Mater* **6**, 173 (2020).

(15) Drax, C. & Scheffler, M. The NOMAD laboratory: from data sharing to artificial intelligence. *J. Phys. Mater.* **2**, 036001 (2022).

(16) Pizzi, G., Cepellotti, A., Sabatini, R., Marzari, N. & Kozinsky, B. AiiDA: automated interactive infrastructure and database for computational science. Comp. Mater. Sci. **111**, 218–230 (2016).

(17) Talirz, L., Kumbhar, S., Passaro, E. et al. Materials Cloud, a platform for open computational science. *Sci Data* **7**, 299 (2020)

(18) Time for lithium-ion alternatives. *Nat Energy* **7**, 461 (2022).

(19) Chu, S., Cui, Y. & Liu, N. The path towards sustainable energy. *Nature Mater* **16**, 16–22 (2017).

(20) Grey, C.P., Hall, D.S. Prospects for lithium-ion batteries and beyond—a 2030 vision. *Nat Commun* **11**, 6279 (2020).

(21) Biemolt, J., Jungbacker, P., van Teijlingen, T., Yan, N., & Rothenberg, G. Beyond Lithium-Based Batteries. *Materials (Basel)*, **13**, 2, 425 (2020).

(22) Liu, Q et al. Rechargeable anion-shuttle batteries for low-cost energy storage. *Chem* **7**, 8, 1993-2021 (2021).

(23) Anji Reddy, M. & Fichtner, M. Batteries based on fluoride shuttle. *J. Mater. Chem.* **21**, 17059 (2011).

(24) Nowroozi, M. A. et al. Fluoride ion batteries – past, present, and future. *J. Mater. Chem. A* **9**, 5980–6012 (2021).

(25) Himanen, L., Geurts, A., Foster, A. S., Rinke, P., Data-Driven Materials Science: Status, Challenges, and Perspectives. *Adv. Sci.* **6**, 1900808 (2019).

(26) NIST X-ray Photoelectron Spectroscopy Database, Version 4.1 *National Institute of Standards and Technology* (2012) http://srdata.nist.gov/xps/.

(27) Mehl, M. J. et al. The AFLOW library of crystallographic prototypes: part 1. *Comp. Mater. Sci.* **136**, S1–S828 (2017).

(28) Hicks, D. et al. The AFLOW library of crystallographic prototypes: part 2. *Comp. Mater. Sci.* **161**, S1–S1011 (2019).

(29) Hicks, D. et al. The AFLOW library of crystallographic prototypes: part 3. *Comp. Mater. Sci.* **199**, 110450 (2021).

(30) Andersen, C.W. et al, OPTIMADE, an API for exchanging materials data, *Sci. Data* **8**, 217 (2021).

(31) Li, H., Armiento, R., & Lambrix, P. An Ontology for the Materials Design Domain. *Arxiv* (2020) arXiv:2006.07712.

(32) Sundberg, J.D; Benjamin, S.S.; McRae, L.M.; & Warren, S.C. Simmate: a framework for materials science. *JOSS* **7**, 75, 4364 (2022).

(33) Lee, K., Kim, S. W., Toda, Y., Matsuishi, S., & Hosono, H. Dicalcium nitride as a two-dimensional electride with an anionic electron layer. *Nature* **494**, 7437, 336–340 (2013).

(34) Kitano, M. et al. Essential role of hydride ion in ruthenium-based ammonia synthesis catalysts. *Chem. Sci.* **7**, 7, 4036-4043 (2016).

(35) Kim, S. et al. Long-Range Lattice Engineering of MoTe2 by a 2D Electride. *Nano letters* **17**, 6, 3363–3368 (2017).

(36) Druffel, D.L. et al. Electrons on the surface of 2D materials: from layered electrides to 2D electrenes. *J. Mater. Chem. C* **5**, 43, 11196-11213 (2017).

(37) Zhang, X.; Yang, G. Recent Advances and Applications of Inorganic Electrides. *J. Phys. Chem. Lett.* **11**, 3841– 3852 (2020).

(38) McRae, L.M. et al. Sc2C, a 2D Semiconducting Electride*. J. Am. Chem. Soc.* **144**, 24, 10862-10869 (2022).

(39) Zhao, X., Zhao-Karger, Z., Fichtner, M. & Shen, X. Halide-based materials and chemistry for rechargeable batteries. *Angew. Chem. Int. Ed.* **59**, 5902–5949 (2020).

(40) Gschwind, F. et al. Fluoride ion batteries: theoretical performance, safety, toxicity, and a combinatorial screening of new electrodes. *J. Fluor. Chem.* **182**, 76–90 (2016).

(41) Dieudonné, B. et al. The key role of the composition and structural features in fluoride ion conductivity in tysonite $Ce_{1-x}Sr_xF_{3-x}$ solid solutions. *Dalton Trans.* **46**, 3761–3769 (2017).

(42) Dieudonné, B. et al. Exploring the $Sm_{1-x}Ca_xF_{3-x}$ tysonite solid solution as a solid-state electrolyte: relationships between structural features and F^- ionic conductivity. *J. Phys. Chem. C.* **119**, 25170–25179 (2015).

(43) Preishuber-Pflügl, F., Epp, V., Nakhal, S., Lerch, M. & Wilkening, M. Defect-enhanced F^- ion conductivity in layer-structured nanocrystalline $BaSnF_4$ prepared by high-energy ball milling combined with soft annealing. *Phys. Status Solidi C.* **12**, 10–14 (2015).

(44) Ahmad, M. M., Yamane, Y. & Yamada, K. Structure, ionic conduction, and giant dielectric properties of mechanochemically synthesized $BaSnF_4$. *J. Appl. Phys.* **106**, 074106 (2009).

(45) Sorokin, N. I. SnF_2 -Based Solid Electrolytes. Inorg. Mater. **40**, 989–997 (2004).

(46) Mohammad, I., Witter, R., Fichtner, M. & Anji Reddy, M. Room-temperature, rechargeable solid-state fluoride-ion batteries. ACS Appl. Energy Mater. 1, 4766–4775 (2018).

(47) Nowroozi, M. A., Wissel, K., Rohrer, J., Munnangi, A. R. & Clemens, O. $LaSrMnO_4$: reversible electrochemical intercalation of fluoride ions in the context of fluoride ion batteries. *Chem. Mater.* **29**, 3441–3453 (2017).

(48) Nowroozi, M. A., de Laune, B. & Clemens, O. Reversible electrochemical intercalation and deintercalation of fluoride ions into host lattices with schafarzikite-type structure. *Chem. Open* **7**, 617–623 (2018).

(49) Clemens, O. et al. Electrochemical fluorination of perovskite type $BaFeO_{2.5}$. *Dalton Trans.* **43**, 15771–15778 (2014).

(50) Druffel, D. L. et al. First-principles prediction of electrochemical electron-anion exchange: ion insertion without redox. *J. Phys. Chem. Lett.* **11**, 9210–9214 (2020).

(51) Druffel, D. L. et al. Synthesis and Electronic Structure of a 3D Crystalline Stack of MXene-Like Sheets. *Chem. Mater.* **31**, 23, 9788–9796 (2019).

(52) Martoňák, R.; Laio, A.; & Parrinello, M. Predicting Crystal Structures: The Parrinello-Rahman Method Revisited. *Phys. Rev. Lett.* **90**, 075503 (2003).

(53) Martoňák, R. et al. Simulation of structural phase transitions by metadynamics*. Zeitschrift für Kristallographie – Crys. Mat.* **220**, 5-6, 489-498 (2005).

(54) Pickard, C.J. & Needs, R.J. Ab initio random structure searching. *J. Phys.: Condens. Matter* **23**, 053201 (2011).

(55) Oganov, A.R (Ed.). Modern Methods of Crystal Structure Prediction, *Wiley VCH* (2010).

(56) Lyakhov, A.O. et al. New developments in evolutionary structure prediction algorithm USPEX. *Comp. Phys, Comm.* **184**, 4, 1172-1182 (2013).

(57) Falls, Z. et al. The XtalOpt Evolutionary Algorithm for Crystal Structure Prediction. *J. Phys. Chem. C* **125**, 3, 1601–1620 (2021).

(58) Vilhelmsen, L. B. & Hammer, B. A genetic algorithm for first principles global optimization of supported nano structures. *J. Chem. Phys.* **141**, 044711 (2014).

(59) Revard, B. C.; Tipton, W. W.; & Hennig, R. G. GASP: Genetic algorithm for structure and phase prediction. *Online Github Repository* https://github.com/henniggroup/GASP-python (2018).

(60) Wang, Y.; Lv, J.; Zhu, L.; & Ma, Y. CALYPSO: A method for crystal structure prediction. *Comp. Phys. Comm.* **183**, 10, 2063-2070 (2012).

(61) Liu, X., Niu, H. & Oganov, A.R. COPEX: co-evolutionary crystal structure prediction algorithm for complex systems. *npj Comput Mater* **7**, 199 (2021).

(62) Scott, E.O. & De Jong, K.A. Understanding Simple Asynchronous Evolutionary Algorithms. *FOGA* 85–98 (2015).

CHAPTER 2: CORE CONCEPTS AND WORKFLOWS

2.1 Introduction

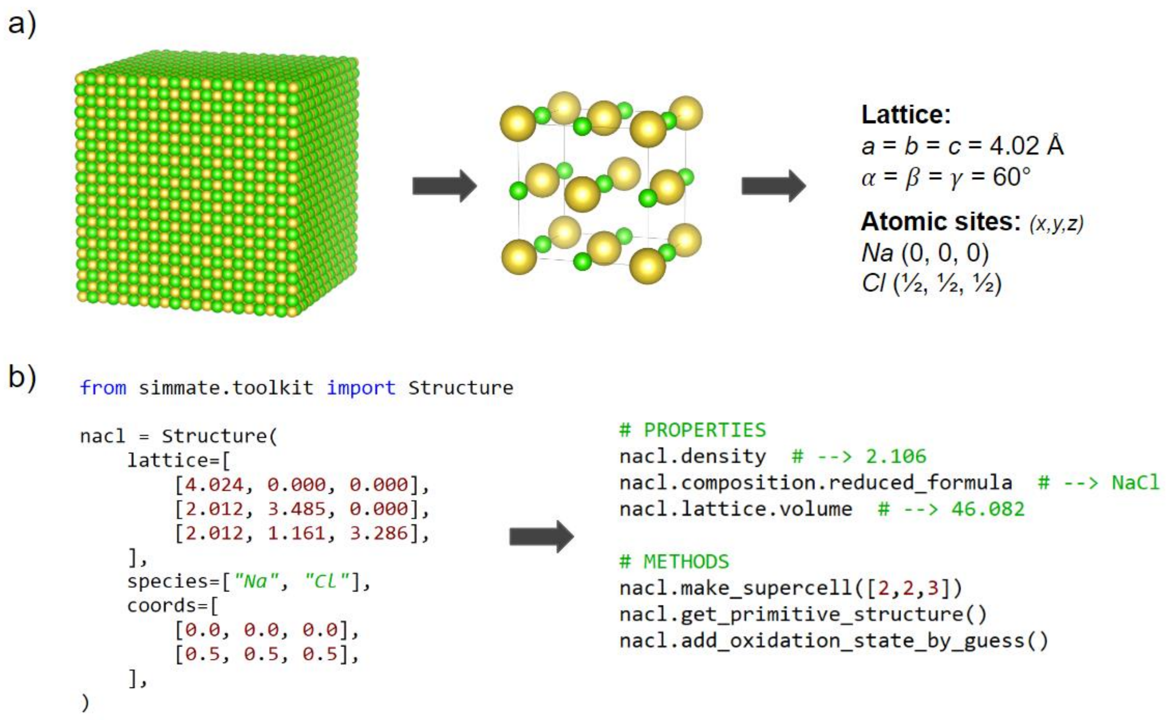
Big data and high-throughput calculations have been growing in popularity in material science and – during the course of my Ph.D. program – have made their way into the Warren Lab at UNC. Many new computational capabilities were introduced to our lab during this time, and now each provides essential insights for our experimental work. These new methods range from the adoption of third-party software as well as the development of our own alternatives.

This chapter is dedicated to the underlying concepts and methods that make our computational research possible, while Chapters 3-6 focus on the high-level frameworks and materials chemistry achieved by applying these concepts and methods. First, we introduce material informatics concepts that are essential for maintaining and contributing to our existing code bases, e.g., how data is organized, tested, and transferred. Next, we cover how data is generated – i.e., how material properties are predicted through common workflows and underlying ab initio methods. Additional in-depth guides and details are available within the respective chapters and (more generally) our online documentation.

2.2 Material informatics and data science

2.2.1 Python class ontology

Materials science ontology (i.e., the many ways we can define and organize materials science data) helps to connect related information and analyses.[1] At the lowest level, we organize data types in Python into "classes" and "objects". In real life, acetone, acetonitrile, and ethanol are examples of solvents. In Python, we say that *acetone*, *acetonitrile*, and *ethanol* are objects of the *Solvent* class. By organizing objects into classes, Python simplifies the way we program. For example, we could design the *Solvent* class to have a property called *boiling_point*. Then, we could view the property simply by typing "*acetone.boiling_point*". We (essentially) set rules that, no matter what *Solvent* we have, its properties can be accessed with "*example_solvent.example_property*". This might seem silly, but it becomes very powerful once we want to start building out new functionality and features.



```python
from simmate.toolkit import Structure

nacl = Structure(
    lattice=[
        [4.024, 0.000, 0.000],
        [2.012, 3.485, 0.000],
        [2.012, 1.161, 3.286],
    ],
    species=["Na", "Cl"],
    coords=[
        [0.0, 0.0, 0.0],
        [0.5, 0.5, 0.5],
    ],
)
```

```python
# PROPERTIES
nacl.density  # --> 2.106
nacl.composition.reduced_formula  # --> NaCl
nacl.lattice.volume  # --> 46.082

# METHODS
nacl.make_supercell([2,2,3])
nacl.get_primitive_structure()
nacl.add_oxidation_state_by_guess()
```

Figure 2-1. Introduction to the Python Structure class. (a) An example bulk crystal structure for sodium chloride is shown (left), along with its conventional unit cell (center), and symmetrically reduced primitive cell parameters (right). Instead of a graphical interface, (b) the *Structure* class embeds the same crystallographic information in Python (left), which then allows programmatic calculation of properties as well as manipulations of the crystal (right).

In materials science, the class we use most is for crystal structures. We call this the *Structure* class and require that all *Structure* objects be made of a lattice and a list of atomic sites (Figure 2-1). Once the lattice and sites are known, the *Structure* class allows us to automate the calculation of properties, perform complex analyses, or transform the structure in new structures. In the simplest cases, this could be calculating the volume, density, or distances between all sites. In more advanced cases, the many smaller functions can build up to complex symmetry conversions, diffusion analysis, or evaluation of coordination environments.[2,3] Together, there are many properties and methods available for objects from the *Structure* class – all built from our simple definition of '*structure = lattice + sites*'.

Classes are also available for many other common data types, such as *Lattice*, *PeriodicSite*, *Molecule*, *Element*, *Ion*, *Bond*, *ElasticTensor*, and many others. Moreover, similar functionalities may be grouped into their own classes – e.g., there is the *Transformation* class that constitutes some modification to *Structure* objects (such as a site perturbation). It can become daunting to see how many classes are available, but by organizing our code into classes with distinct rules, it becomes easier for others to add new features. This is most apparent with the aforementioned *Transformation* class. We can have a diverse set of methods that modify a *Structure* object (e.g., *CubicSupercellTransformation* or *GrainBoundaryTransformation*), but all instances share a common "*apply_to_structure*" method that is straightforward to define and use. Thus, the primary goals of Python classes are to (i) organize our code and functionality and (ii) prepare for the addition, replacement, and modification of new features.

Materials science ontology varies greatly depending on the topic being studied, which has ultimately led to the development of several Python ecosystems. The most popular of these are the Atomic Simulation Environment (ASE)[2] and the Python Package for Material Genomics (PyMatGen)[3], both of which have been actively developed since 2008. Many other software packages borrow concepts from and build off ASE and PyMatGen ontologies – giving rise to packages for various subdisciplines in materials science.[4,5,6,7,8,9,10,11]

The software introduced in this dissertation is no exception. We frequently borrow ontologies and even use many third-party functions (i.e., code and utilities from other software) [2,3,12,13] directly within our code. This established a reliable starting point by which we could grow out higher-level, advanced features, such as a workflow library and full feature database. Our ontology is constantly changing and expanding as we incorporate new areas of study, but the core classes (*Structure*, *Workflow*, *DatabaseTable*, and *WebsiteView*) serve as the foundation of higher-level functions.

2.2.2 Continuous-integration and maintenance

As Python classes become interconnected, one small change or typo can easily cascade into larger problems elsewhere. For example, if we change the name of a method in the *PeriodicSite* class, methods of the *Structure* class which uses *PeriodicSite* will fail as a result. Developers must therefore constantly check that their code works as expected, even when a single line of code is changed. Developers establish a series of tests and checks to be run whenever a change is made, and the automation of this practice is known as continuous integration (CI).

The CI framework consists of (i) unit tests, (ii) code coverage, (iii) code linting, and (iv) documentation. All parts are automated through GitHub CI, an open-source platform for maintaining code.

Each component helps maintain code quality and reproducibility, which in turn greatly influence the code's longevity and influence in the field.[14]

The most important aspect of CI is the suite of unit tests. Here, a "unit test" runs a single function and ensures that the output is the expected one. For example, a simple test is to load NaCl from a file and confirm that calling "*nacl.volume*" returned the expected volume. For all of the software presented in this dissertation, we implemented over 700 unique tests that range from accessing structure properties to testing multicomponent workflows/webpages. Within Github CI, tests are implemented using PyTest[15], which uses several plugins for database management[16], command-line interface (CLI) emulation[17], and test parallelization[18]. Every time a change is made in the code, the test suite determines whether any features were broken by the change.

In addition, code may behave differently on various operating systems or versions of Python. CI must ensure that our code works for each configuration. We therefore maintain a matrix of common user configurations (e.g., different OSs × different Python versions), and the GitHub CI automatically runs our unit test suite with each configuration.[19] We also restrict supporting programs (i.e., package dependencies) in Python configuration files such as "*pyproject.toml*". Together, these test features limit users to environments that are confirmed to work and facilitate tracking issues that are unique to specific setups.

Unit tests should evaluate as many of our features as possible. Ideally, all code should be tested, but this isn't always possible in practice due to code complexity, computational costs, and complex configurations (i.e., emulating multi-machine setups). To quantify the amount of code that is run during all unit tests, Github CI calculates the percent of lines of code that are tested. This is known as coverage. It is common for peer-reviewed packages to be published with ~50-75% coverage.[15,20,21] At the time of publication, our software (presented in Chapter 3) was at 84% coverage, and its coverage fluctuates as new features and/or tests are added. As new code is added, it is essential to monitor coverage and determine where additional unit tests should be implemented.

Code linting, that is code style and conventions, ensures code is readable and clean.[14] The Python community has developed a series of rules to ensure all code is readable to all Python programmers. Code linting is implemented through the Black[22] and iSort[23] Python packages. We follow this standard, so that our code will automatically conform to best practices.

The final CI component is documentation. Documentation is hosted on an independent server that includes tutorials, guides, and API reference. In earlier versions, we relied on pDoc[24] to automatically generate a documentation website because this package encouraged users to open and read the source code. However, as our software package expanded, we found that code vs. guide organization needed to be handled separately. For this reason, we currently use MkDocs[25] and MkDocStrings[26], which automatically build documentation from markdown text files and source code. By integrating MkDocs into the Github CI, the website is rebuilt and redeployed after each change is accepted into the code.

2.2.3 Class abstraction and data conversion

Even though the definitions of classes such as *Structure* or *Molecule* are straightforward, there are a surprisingly large number of ways to store, format, and manipulate instances of this data. Moreover, scientific groups tend to adopt a particular format and develop all their software and utilities around it.[27,28] Ultimately, this means data sharing between different programs – such as exploring multiple crystal databases – becomes nontrivial and frequently forces users to manually carry out such steps.[29]

To address this, base classes (e.g., *Structure*) provide a common entry point that diverse formats and software can connect to. This is known as input-output conversion (IO) for file formats and external function wrapping (EXT) for integrating other programs' functionality. IO and EXT methods implement a two-step process for users to follow (Figure 2-2). Users (i) indicate the format of input data (e.g., a structure file, calculation, database entry, API, etc.) and (ii) immediately gain access to all integrated data and functions (e.g., different *ab initio* software).

Python's ability to integrate many different formats and function is particularly powerful. The *Structure* class currently supports IO for VASP (POSCAR), CIF, LAMMPS, and >110 other formats through OpenBabel.[30] It can also jump between structure-like classes from other packages (e.g., PyMatGen[3], ASE[2], JARVIS[31], and SQL database models[27]). This means that Python methods can switch to and from these formats and automate external analyses that rely on them. For example, we can use EXT to easily access many different programs for structure creation.[2,33,34,35,36,37,38] Although different formats and functions are used across these programs (Table 2-1), we can make them all compatible.

Regardless of the original program and format, users can always generate new structures with a "*Creator.new_structure()*" method and receive the result as a *Structure* object.

Python's ability to unify disparate database is especially important. Each large large-scale database (Materials Project[39], AFLOW[40], JARVIS[31], COD[28], and OQMD[41]) operates under different ontologies and interfaces, which previously made studies that utilize more than one database exceedingly rare. However, by using the IO and EXT concepts from above, data from many different providers can be unified under a single interface.

In summary, these examples illustrate how core Python classes can bridge the many features and formats developed by the community – making them all available under a common interface.

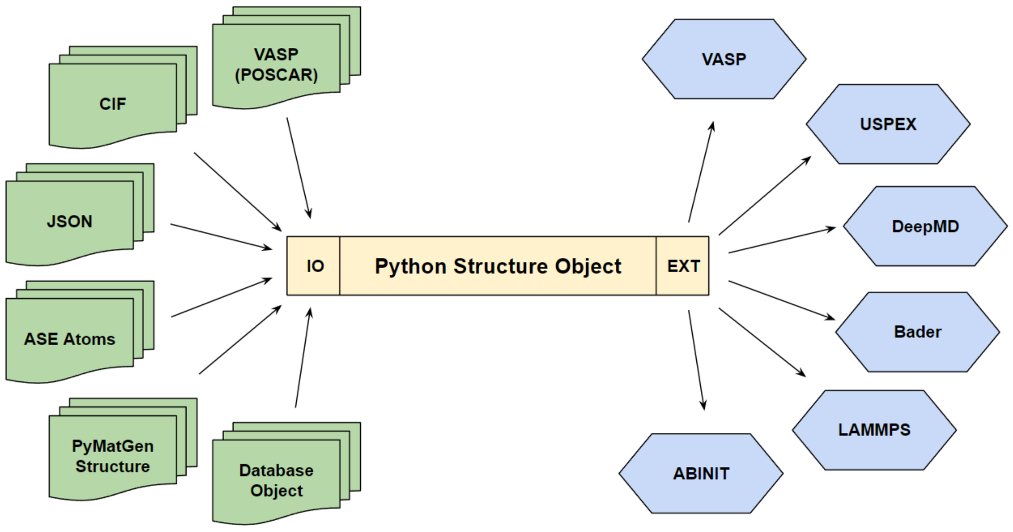


Figure 2-2. Overview of the Structure class IO/EXT process. Crystal structures can come from a variety of input formats (left) that can be converted to a Python *Structure* object(s) (center). The *Structure* object can be manipulated using core methods and then returned to external programs in their desired file formats (right).

Table 2-1. Programs for the creation of "random" crystal structures. Each program's required file format is shown to illustrate how Structure IO facilitates program integration.

Software / Package	**File Format**	**Interface(s)**	**Supported in our code?****
AIRSS[37]	CIF	CLI	Yes
ASE[2,34]	ASE Atoms	Python	Yes
CALYPSO[36]	POSCAR	CLI	Yes
GASP[35]	PyMatGen Structure	Python	Yes (requires custom patch)
PyXtal[38]	PyMatGen Structure	CLI or Python	Yes
USPEX[32]	POSCAR	CLI	Yes
XtalOpt[33]	POSCAR	CLI	Yes (uses randSpg directly)

*** These are part of the software package that is introduced in Chapter 3.*

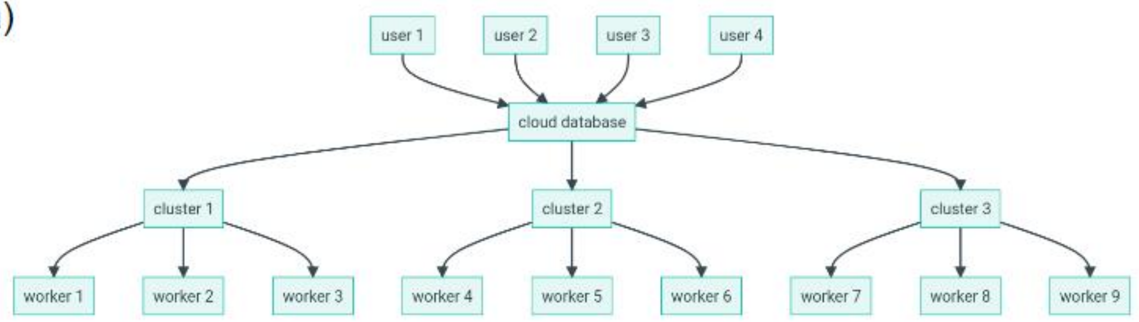
2.2.4 Distributed computational resources

Calculations should efficiently scale across all available resources to speed up an analysis. These resources could be a single laptop, a collection of desktop personal computers (PCs), commercial cloud servers, university high performance computing (HPC) clusters, or some combination of these options. Unfortunately, each resource typically has a different networking protocol (e.g., secure shell protocol (SSH)) and queue system (e.g., SLURM or PBS).[42] To address this challenge, our informatics framework is built to communicate entirely through a centralized cloud database. Each resource is added by calling a '*start-worker*' command, where a worker is defined as a resource that runs scheduled workflows. This means that a new resource can be started on any computer connected to the internet, does not require any prior knowledge of scheduled workflows, and can be added/removed at any point without affecting other resources, workflows, or analyses. Thus, our framework allows complex analyses to run on a single computer while also quickly scaling across massively parallel resources.

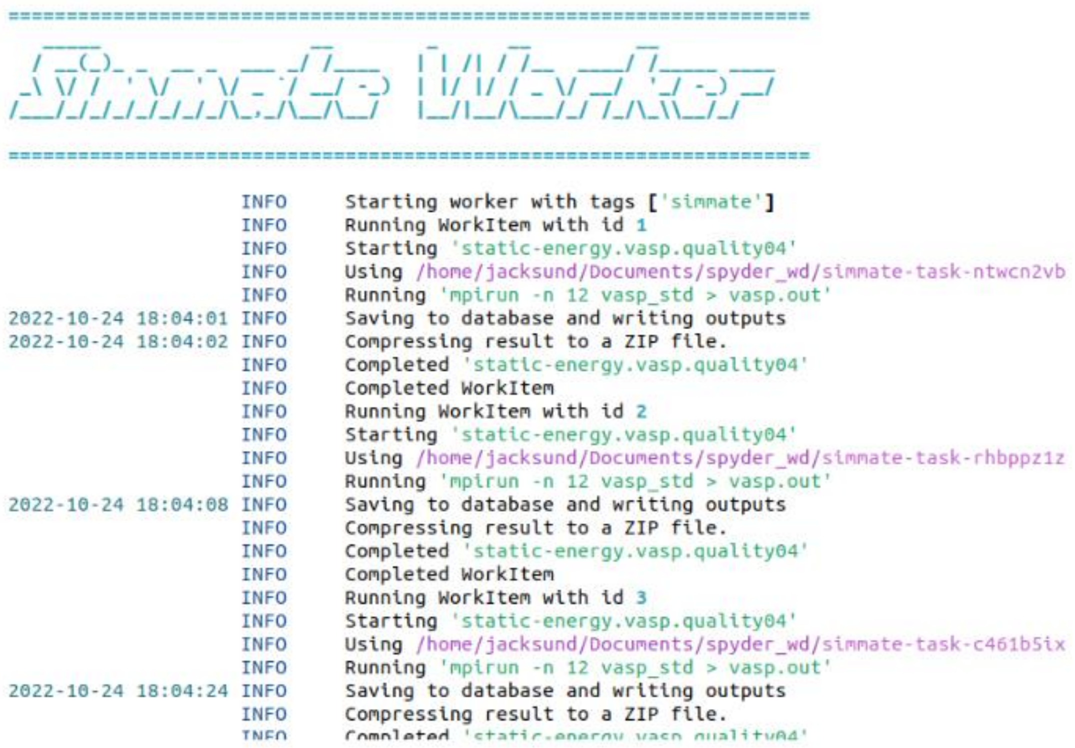
In practice, we have used this architecture to run up to 100,000 calculations per day across all of UNC's available HPC clusters (LongLeaf, DogWood, KillDevil, WarWulf, etc.) and automatically feed the results back to a central database for analysis. As our calculations continue to grow in size, we anticipate this will scale to include all of the lab's desktops and even HPC clusters from collaborating universities. Importantly, different resources can be subgrouped (or "tagged") for specific types of workflows, which ensures that work is run by optimal resources and that shared resources are not overused.

Additional advantages of this architecture become apparent through example use cases. These cases are demonstrated in chapters 4-6. All chapters illustrate massive parallel high-throughput searches and analyses while also illustrating how cross-resource data sharing accelerates analyses.

a)



b)



```
=====================================================================
   _____            __        _      __         __
  / __(_)_ _  __ _  ___ _/ /____  | | /| / /__  ____/ /_____ ____
 _\ \/ /  ' \/  ' \/ _ `/ __/ -_) | |/ |/ / _ \/ __/  '_/ -_) __/
/___/_/_/_/_/_/_/_/\_,_/\__/\__/  |__/|__/\___/_/ /_/\_\\__/_/

=====================================================================

                     INFO     Starting worker with tags ['simmate']
                     INFO     Running WorkItem with id 1
                     INFO     Starting 'static-energy.vasp.quality04'
                     INFO     Using /home/jacksund/Documents/spyder_wd/simmate-task-ntwcn2vb
                     INFO     Running 'mpirun -n 12 vasp_std > vasp.out'
2022-10-24 18:04:01 INFO     Saving to database and writing outputs
2022-10-24 18:04:02 INFO     Compressing result to a ZIP file.
                     INFO     Completed 'static-energy.vasp.quality04'
                     INFO     Completed WorkItem
                     INFO     Running WorkItem with id 2
                     INFO     Starting 'static-energy.vasp.quality04'
                     INFO     Using /home/jacksund/Documents/spyder_wd/simmate-task-rhbppz1z
                     INFO     Running 'mpirun -n 12 vasp_std > vasp.out'
2022-10-24 18:04:08 INFO     Saving to database and writing outputs
                     INFO     Compressing result to a ZIP file.
                     INFO     Completed 'static-energy.vasp.quality04'
                     INFO     Completed WorkItem
                     INFO     Running WorkItem with id 3
                     INFO     Starting 'static-energy.vasp.quality04'
                     INFO     Using /home/jacksund/Documents/spyder_wd/simmate-task-c461b5ix
                     INFO     Running 'mpirun -n 12 vasp_std > vasp.out'
2022-10-24 18:04:24 INFO     Saving to database and writing outputs
                     INFO     Compressing result to a ZIP file.
                     INFO     Completed 'static-energy.vasp.quality04'
```

Figure 2-3. Setup for distributed computational resources. (a) All users, workflows, and resources are connected to a cloud database, which manages all scheduled calculations. Clusters vary in complexity from a laptop with one CPU to an entire HPC system with thousands of CPUs (e.g., WarWulf, DogWood, LongLeaf, KillDevil, etc.). Workers are given access to a subset of the cluster's resources (e.g., two threads of a laptop's CPU or 2000 threads of a cluster) to perform a workflow. On a cluster, workers are created and run within the cluster's job scheduler, such as SLURM. (b) An example log file for a single worker, which shows it working through a series of workflows.

2.2.5 Data serialization and transfer

Because our calculations are distributed across separate computer networks, we must be able to efficiently transfer data among them. There are many strategies to do this, ranging from creating a shared file system to sending bits of data through an internet connection. Furthermore, data must first be packaged in a machine-readable format before it can be transferred. This process is known as serialization, and it greatly affects transfer speed.[43] For example, if we have a *Structure* object that needs to be sent to a remote resource, we (i) convert that object to a condensed format, (ii) send the condensed data to the new resource, and (iii) rebuild a matching *Structure* object from the condensed data. This process is closely related to the general process of data conversion (section 2.2.3); here, we must carefully select which data to convert and send for our use case. Regardless of the method, both serialization and transfer must be fast, scalable, and robust.

Currently, we implement serialization through Monty[44], a package maintained by the PyMatGen team, where Python objects are written to/from a JavaScript Object Notation (JSON) format. For classes and objects where JSON serialization has not been configured, we use cloudpickle[44] to handle complex objects and prevent data loss. Looking forward, serialization must be optimized in scenarios with large, complex objects such as *BandStructure* or *DensityOfStates*, where serialization and transfer can become rate limiting. Optimization can be achieved by condensing the data (e.g., using JSON) or by using an alternative serialization engine such as MessagePack[46]. However, until serialization becomes rate-limiting, JSON and cloudpickle are effective and preferred.

Once serialized, we can address how data is physically transferred. Step (ii) above suggests that data is sent directly from one resource to another (i.e., from a local PC to a remote cluster); however, in practice, we use an intermediate resource that is dedicated entirely to data transfer and distribution. A user would send their data to this single server, which then handles transfer to all other resources. Much like a single base class simplifies data conversion (Figure 2-2), a single endpoint for data transfer simplifies how resources connect and communicate with one another (Figure 2-3).

There are several options for this central data store: in-memory databases, Structured Query Language (SQL) databases, or a web API. For example, software such as Redis[47] is designed specifically for data orchestration and data caches, serving as an in-memory database from which resources can rapidly write/read data. In-memory servers, however, lose data when a resource crashes. Therefore, we implement data transfer directly through our SQL database, which saves data to disk and is thus recoverable after a resource failure. SQL databases are the most robust and efficient way to orchestrate data transfer, but a case can be made against their required setup. Specifically, such databases require a direct user connection and (in most cases) installed software to access much of the API. For this reason, web APIs such as REST (representational state transfer) or GraphQL (graph query language) are frequently implemented. In simple terms, these APIs are how we can access databases from a website URL – no installations are required beyond a common web browser such as Chrome or FireFox. The API maintains a single database connection and serves JSON-serialized data to authenticated users. While we prefer direct SQL connections in most cases, our software also provides a REST API for those who want to programmatically access data without any software dependencies.

The centralized cloud storage of serialized data allows us to share data across remote resources in a simple and efficient manner. There are many options on how to implement each of these components, and optimal choices depend on the desired function. For now, we heavily use JSON serialization and direct SQL connections, but it is important to keep their alternatives in mind as software demands grow and change.

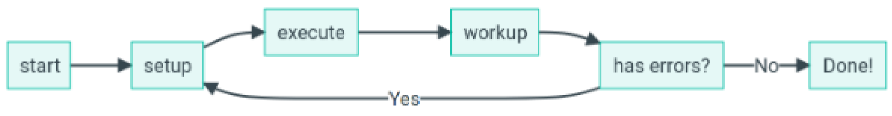
2.3 Workflows for predicting material properties

2.3.1 The workflow engine

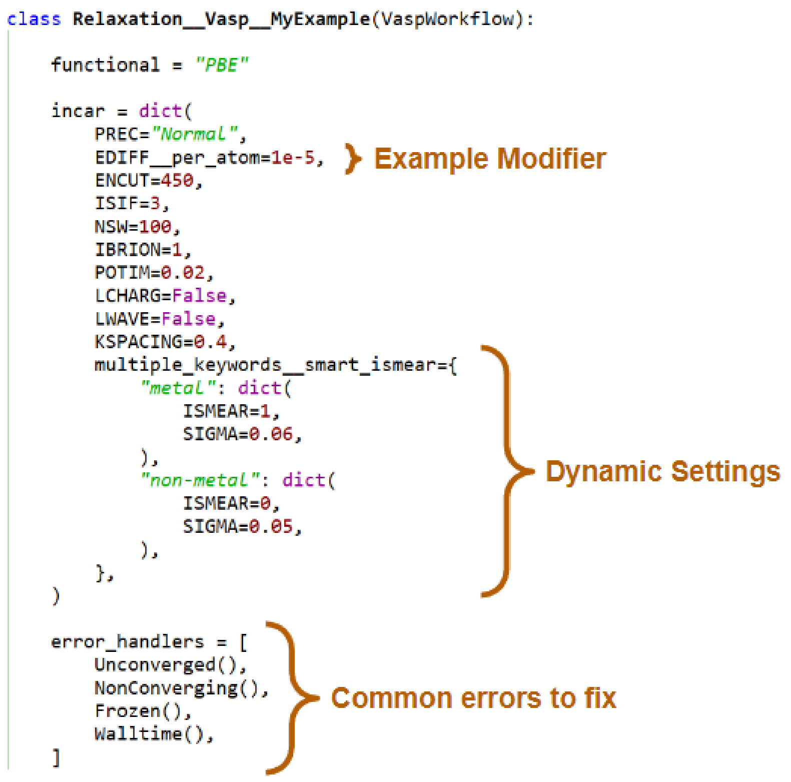
The calculation of material properties frequently follows a set of steps and rules, and these procedures can be automated to enable the calculation of thousands of materials. Many workflow libraries have been built within the materials science community to achieve exactly this. Some are specific to partner software (such as AbiPy[48] for ABINIT[49]), while others are built for more general packages (e.g., Atomate[50] for VASP[51], and LAMMPS[52]). Regardless of the program they use, all workflow packages build out a workflow engine by which they can define and orchestrate their workflows.

Workflows are defined by grouping units of work into 'tasks' and then deciding how these tasks should be distributed to the available resources. Existing workflow engines (JobFlow[53], FireWorks[54], AirFlow[55], etc.) implement workflows as directed acyclic graphs (DAGs), where tasks are connected in a specific direction and individual steps never repeat. This rigidity greatly restricts the manner in which workflows can be defined and submitted. Instead, our framework adopts DAG-free tasks where task connectivity is not required so that any Python logic is allowed. This strategy is implemented through the use of newer workflow engines and task executors such as Prefect[56] and Dask[57]. Here, any Python code is wrapped with a single class method, so developers can decide whether to define tasks. This also allows workflows to be used within one another (i.e., nested workflows). The DAG-free framework leads to pythonic workflows that have total control over logic and orchestration of individual tasks.

a)



b)



```
class Relaxation__Vasp__MyExample(VaspWorkflow):

    functional = "PBE"

    incar = dict(
        PREC="Normal",
        EDIFF__per_atom=1e-5,
        ENCUT=450,
        ISIF=3,
        NSW=100,
        IBRION=1,
        POTIM=0.02,
        LCHARG=False,
        LWAVE=False,
        KSPACING=0.4,
        multiple_keywords__smart_ismear={
            "metal": dict(
                ISMEAR=1,
                SIGMA=0.06,
            ),
            "non-metal": dict(
                ISMEAR=0,
                SIGMA=0.05,
            ),
        },
    )

    error_handlers = [
        Unconverged(),
        NonConverging(),
        Frozen(),
        Walltime(),
    ]
```

Figure 2-4. Overview of dynamic S3Workflows. (a) A supervised-staged-shell (S3) workflow runs a file-based program while monitoring and fixing common errors until the run completes successfully. S3Workflows are used to build workflows for VASP, where an example workflow (b) can dynamically control input settings and errors to watch for. Many more features are available with these workflows.

Pythonic workflows allow us to leverage class-based inheritance for common functionality such as calling an external program, handling input/output files, and monitoring files for common errors or issues. Our framework includes a base *supervised-staged-shell* workflow (*S3Workflow*) to handle this common process (Figure 2-4a). *S3Workflow*s automate writing inputs, monitoring jobs, fixing errors, and loading output files back into Python through a single modular interface. To illustrate its utility, we have fully reimplemented many of the Materials Project's presets and >20 error handlers for VASP (checks for convergence issues, incorrect smearing for metals/nonmetals, batch job wall times, etc.), where these features were previously distributed across several packages and modules.[3,39,50,58] By integrating all of this into a *S3Workflow*, we were able to develop over 50 unique and dynamic workflows and establish a toolkit to create additional custom workflows with minimal effort.

2.3.2 Density functional theory settings

All density functional theory (DFT) calculations were performed with the Vienna ab initio Simulation Package (VASP)[51]. Projector-augmented wave (PAW) potentials were used to describe core electrons, while the generalized gradient approximation (GGA) Perdew–Burke-Ernzerhof (PBE) functional was used for the exchange-correlation contribution to total energy.[59,60] Furthermore, Grimme DFT-D3 and Hubbard U corrections were used to account for long-range dispersion forces and local/semi-local deficiencies, respectively, where noted.[61,62,63] Alternative functionals (e.g., LDA[64], SCAN[65], HSE[66]) are also implemented and frequently used with our workflow framework.

General settings (e.g., for k-point grids, convergence criteria, and smearing partial occupancies) depended on both the type of analysis being performed and the structure itself. For example, many relaxations were performed at varying quality in different analyses, such as lower quality settings when analyzing tens of thousands of structures and higher quality settings once promising candidates were identified. This ultimately led to >50 unique workflow presets, including >10 presets for ionic relaxations alone. Moreover, within a single preset, many settings/corrections were dynamically determined from preconfigured defaults, e.g., specific compositions and elements would have U values automatically set. We therefore encourage readers to view the full configurations by directly interacting with our software, either through the CLI or source code (Figure 2-4b).

2.3.3 Ionic relaxations

Crystal structures are often optimized using a series of increasingly strict tolerances, which we have automated. For example, our evolutionary search algorithm relies on rapid rough relaxations of randomly generated (and often highly unreasonable) structures, so a series of four relaxations and one final static energy calculation are performed for each structure. The first relaxation uses a fixed volume and a 0.75 $Å^{-1}$ Monkhorst-Pack k-point grid, where a conjugate-gradient algorithm is applied with convergence criteria of 2.0×10^{-2} eV/atom for energy, and 1.0×10^{-2} eV/Å for max force. Then, the convergence settings are gradually increased up to the final relaxation, which uses an RMM-DIIS algorithm[67], 1.0×10^{-5} eV/atom for energy, 1.0×10^{-3} eV/Å for max force. Follow-up calculations and analyses relax structures using higher quality Materials Project and MIT project settings.[3,29]

2.3.4 Population Analysis

Oxidation state analysis – i.e., the assignment of electron density populations to individual atoms – greatly affects our interpretation of electride systems and their analogs. However, the presence of lone electrons in electrides greatly complicates the prediction of oxidation states. We therefore carried out multiple analyses using a series of partitioning algorithms, and we also outline other potential techniques that are worth investigating.

Oxidation states of ionic materials are most often determined via Bader analysis[68], where charge density is partitioned using zero-flux surfaces. For electrides, Bader analysis requires the introduction of “dummy” atoms located at the maxima of electride density to properly assign populations.[69] However, due to the low total electron density in electride regions (relative to atomic sites), Bader analysis struggles to identify zero flux surfaces around these known electride sites. This ultimately leads to underestimation of electride character and overestimation of the e^- density on neighboring atoms (Table 2-2, red).

For this reason, electrides are typically evaluated using the electron localization function (ELF).[70] Here, voxel quantities correspond to the probability of finding an electron of the same spin as a reference electron in a region. In other words, smaller probabilities correspond to an electron that is already localized and stable in a region of space. Localization probabilities vary greatly between crystal structures, but by

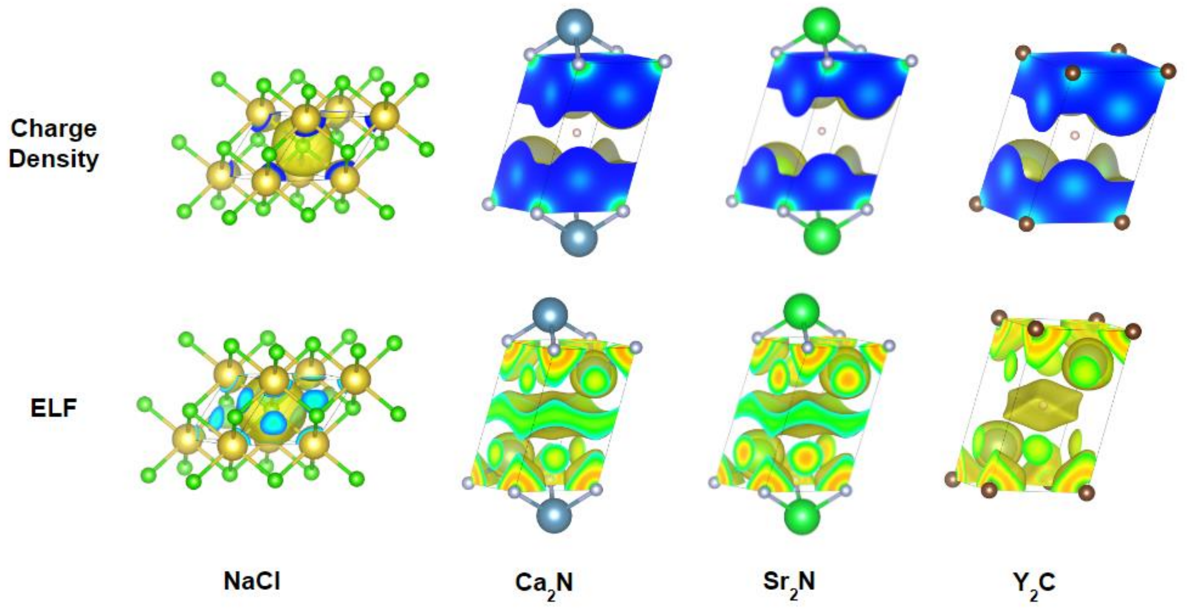


Figure 2-5. Charge density and ELF isosurfaces of example ionic and electride structures. Charge densities (top) often fail to partition electride electron density in layered electrides, whereas ELFs (bottom) show a clear region when electride electron localization occurs. ELF data can be further used to assign charge densities in electrides (see Table 2-2).

Table 2-2. Oxidation state assignments via Bader and Bader/ELF methods. Columns correspond to the oxidation states assigned to composition elements (left to right, in the same order they appear in the chemical formula).

	Bader Analysis			Bader Analysis w. ELF reference		
NaCl	+0.867	-0.867	--	+0.965	-0.965	--
$Ca_2N(e^-)$	+1.005	-2.011	0	+1.694	-2.399	-0.988
Ca_2NF	+1.411	-1.928	-0.895	+1.678	-2.415	-0.941
Ca_2NCl	+1.401	-1.947	-0.856	+1.693	-2.434	-0.951
$Sr_2N(e^-)$	+0.968	-1.933	0	+1.683	-2.395	-0.971
Sr_2NF	+1.408	-1.918	-0.898	+1.680	-2.426	-0.935
Sr_2NCl	+1.396	-1.933	-0.858	+1.692	-2.439	-0.946
$Y_2C(e^-)_2$	+1.263	-2.270	-0.256 (total)	+2.308	-3.041	-1.576 (total)
Y_2CF_2	+1.941	-2.202	-0.839	+2.386	-3.035	-0.868
Y_2CCl_2	+1.841	-2.186	-0.748	+2.374	-3.006	-0.871

examining ELF isosurface values, one can arrive at a qualitative representation of electride character (Figure 2-5).

We sought to introduce a new partitioning strategy that addressed the shortcomings of Bader analysis and ELF visualization. Specifically, the Bader algorithm is used to find zero-flux surfaces in the ELF (instead of charge density), and then the surfaces are used to partition charge density. We found that this approach effectively captures the expected oxidation states in both common ionic materials and electrides (Table 2-2). We do note, however, that the oxidation states of ionic materials (e.g., NaCl) are slightly overestimated relative to Bader analysis.

Alternative strategies to evaluate electride character are also actively being explored. This includes common methods such as partitioning the projected density of states (PDOS) using predicted ionic radii from Bader analysis.[71] This approach, however, is extremely sensitive to changes in radii, so inaccuracies from Bader analysis (which are significant in electride systems) are a major source of concern. Alternatively, one can evaluate chemical bonding using crystal orbital overlap populations (COOP) in newer programs such as LOBSTER.[72] LOBSTER is currently unable to add empty atoms for electrides but support for electride materials is being actively developed.

2.3.5 Ionic Mobility

One can predict ionic mobility using many different computational techniques. Common approaches include nudged elastic band (NEB) and molecular dynamics (MD), and each technique can be implemented with either *ab initio* or empirical potentials. Within this dissertation, we focus on the *ab initio* calculation of NEB barriers because of their high accuracy. [73]

In many cases, we also made approximations to our NEB analysis, such as small supercells, reduced images, and/or partial relaxation of endpoints. For example, our high-throughput analysis includes midpoint-only NEB calculations where endpoints are roughly relaxed. This meant that the predicted barriers were only the minimum possible barrier for each pathway. This is because the true transition state may lie off the midpoint, leading to a higher barrier than what is calculated here. Thus, many of our approximations can be viewed as an extension of Trottier et al.'s explicit-error approach[74] combined with empirical pre-relaxations utilized by Smidstrup et al[75].

However, for the most promising materials, full NEB analyses were performed. This included large supercells (R=10 Å), seven pathway images, and Materials Project convergence settings. All symmetrically pathways in a structure were evaluated, where each unique pathway was identified using pymatgen-diffusion's distinct path finder[3] with two criteria: (1) pathways were limited to 5 Å in length, and (2) only the five shortest symmetrically unique pathways per structure were considered.

2.3.6 Electronic Structure

Band structure (BS) and density of states (DOS) calculations are frequently performed to gain insight into a crystal's electronic structure. Our framework includes workflows for calculating band structures using different levels of theory and corresponding VASP functionals. Specifically, BS and DOS workflows exist for LDA[64], PBE[60], SCAN[65], and HSE[66] functionals. Each of these workflows replicate settings used by Atomate.[50] Although these many workflows exist for quick use on a crystal structure, all high-throughput electronic structure calculations were run using PBE, which is known to systematically underestimate band gaps.[76] We selected this workflow because the high-throughput nature of our calculations prevents higher quality calculations using SCAN or HSE.

2.3.7 Additional workflows

Beyond relaxations, population analysis, ionic transport, and electronic structure calculations, there are many more workflows implemented in the Simmate code. This includes configurations for the calculation of elastic tensors and NMR shifts, as well as large-scale workflows for machine-learned potential creation (using DeePMD[77]), cluster expansion models (using CLEASE[78]), and evolutionary structure prediction (see Chapter 6). It is outside the scope of this work to cover all 50+ workflows and their use cases, so we encourage readers to reference the Simmate documentation for more information.

REFERENCES

(1) Li, H., Armiento, R., & Lambrix, P. An Ontology for the Materials Design Domain. *Arxiv* (2020) arXiv:2006.07712.

(2) Larsen, A. H. et al. The atomic simulation environment—a Python library for working with atoms. *J. Phys.: Condens. Matter* **29**, 273002 (2017).

(3) Ong, S. P. et al. Python Materials Genomics (pymatgen): A robust, open-source python library for materials analysis. Comp. Mater. Sci. **68**, 314–319 (2013).

(4) Evans et al.. Matador: a Python library for analysing, curating and performing high-throughput density-functional theory calculations. *JOSS* **5**, 54, 2563 (2020).

(5) Mosquera-Lois, I., Kavanagh, S. R., Walsh, A., & Scanlon, D. O. Identifying the ground state structures of point defects in solids. *arXiv* **2207**.09862 (2022).

(6) Chen, C. & Ong, S.P. A Universal Graph Deep Learning Interatomic Potential for the Periodic Table. *arXiv* **2202**.02450 (2022).

(7) Baird, S.; Jablonka, K.; & Sayeedm H.M. sparks-baird/xtal2png: v0.9.4 *Zenodo* (2022).

(8) George, J. et al. Automated Bonding Analysis with Crystal Orbital Hamilton Populations. *ChemPlusChem* **87**, e202200123 (2022).

(9) Riebesell, J. et al. Pymatviz: A toolkit for visualizations in materials informatics. *Online GitHub Repository* https://github.com/janosh/pymatviz (2022).

(10) Newville, M.. Larch: An Analysis Package for XAFS And Related Spectroscopies. *J. Phys.: Conf. Series* **430**, 012007 (2013).

(11) Cheng, J.; Luo, J.; & Yang, K. Aimsgb: An algorithm and open-source python library to generate periodic grain boundary structures. *Comp. Mat. Sci.* **155**, 92-103 (2018).

(12) Ward, L., Dunn, A., Faghaninia, A., Zimmermann, N. E. R., Bajaj, S., Wang, Q., Montoya, J. H., Chen, J., Bystrom, K., Dylla, M., Chard, K., Asta, M., Persson, K., Snyder, G. J., Foster, I., Jain, A., Matminer: An open source toolkit for materials data mining. *Comput. Mater. Sci.* **152**, 60-69 (2018).

(13) Deng, Z.; Zhu, Z.; Chu, L.; Ong, S.P. Data-Driven First-Principles Methods for the Study and Design of Alkali Superionic Conductors. *Chem. Mater.* **29**, 1, 281–288 (2017).

(14) Trisovic, A., Lau, M.K., Pasquier, T. et al. A large-scale study on research code quality and execution. *Sci Data* **9**, 60 (2022).

(15) The PyTest collaboration. PyTest: makes it easy to write small tests, yet scales to support complex functional testing for applications and libraries. *Online GitHub Repository* https://github.com/pytest-dev/pytest (2022).

(16) The PyTest-Django collaboration. PyTest-Django: A Django plugin for pytest. *Online GitHub Repository* https://github.com/pytest-dev/pytest-django (2022).

(17) The PyTest-Mock collaboration. PyTest-Mock: Thin-wrapper around the mock package for easier use with pytest. *Online GitHub Repository* https://github.com/pytest-dev/pytest-mock (2022).

(18) The Pytest-XDist collaboration. Pytest-XDist: plugin for distributed testing and loop-on-failures testing modes. *Online GitHub Repository* https://github.com/pytest-dev/pytest-xdist (2022).

(19) The Tox collaboration. TOX: Command line driven CI frontend and development task automation tool. Online GitHub Repository https://github.com/tox-dev/tox (2022).

(20) Hilton, M., Bell, J., & Marinov, D. A Large-Scale Study of Test Coverage Evolution*. Assoc. Comp. Mach.* 53–63 (2018).

(21) Batchelder, N. et al. CoveragePy: The code coverage tool for Python. *Online GitHub Repository* https://github.com/nedbat/coveragepy (2022).

(22) Langa, L. et al. Black: The uncompromising Python code formatter. *Online GitHub Repository* https://github.com/psf/black (2022).

(23) Crosley, T.E. et al. iSort: A Python utility / library to sort imports. *Online GitHub Repository* https://github.com/PyCQA/isort (2022).

(24) Hils, M. et al. pDoc: API Documentation for Python Projects. *Online GitHub Repository* https://github.com/mitmproxy/pdoc (2022).

(25) The MkDocs collaboration. MkDocs: Project documentation with Markdown. *Online GitHub Repository* https://github.com/mkdocs/mkdocs (2022).

(26) Mazzucotelli, T et al. MkDocs-Strings: Automatic documentation from sources. *Online GitHub Repository* https://github.com/mkdocstrings/mkdocstrings (2022).

(27) The EMMET Collaboration. EMMET: be a master builder of databases of material properties. *Online Github repository* https://github.com/materialsproject/emmet (2022).

(28) Gražulis, S. et al. Crystallography open database - an open-access collection of crystal structures*. J. Appl. Crystallogr*. **42**, 726–729 (2009).

(29) Andersen, C.W. et al, OPTIMADE, an API for exchanging materials data, *Sci. Data* **8**, 217 (2021).

(30) O'Boyle, N.M., Banck, M., James, C.A. et al. Open Babel: An open chemical toolbox. J Cheminform **3**, 33 (2011).

(31) Choudhary, K., Garrity, K.F., Reid, A.C.E. et al. The joint automated repository for various integrated simulations (JARVIS) for data-driven materials design. *npj Comput Mater* **6**, 173 (2020).

(32) Lyakhov, A.O. et al. New developments in evolutionary structure prediction algorithm USPEX. *Comp. Phys, Comm.* **184**, 4, 1172-1182 (2013).

(33) Falls, Z. et al. The XtalOpt Evolutionary Algorithm for Crystal Structure Prediction. *J. Phys. Chem. C* **125**, 3, 1601–1620 (2021).

(34) Vilhelmsen, L. B. & Hammer, B. A genetic algorithm for first principles global optimization of supported nano structures. *J. Chem. Phys.* **141**, 044711 (2014).

(35) Revard, B. C.; Tipton, W. W.; & Hennig, R. G. GASP: Genetic algorithm for structure and phase prediction. *Online Github Repository* https://github.com/henniggroup/GASP-python (2018).

(36) Wang, Y.; Lv, J.; Zhu, L.; & Ma, Y. CALYPSO: A method for crystal structure prediction. *Comp. Phys. Comm.* **183**, 10, 2063-2070 (2012).

(37) Pickard, C.J. & Needs, R.J. Ab initio random structure searching. *J. Phys.: Condens. Matter* **23**, 053201 (2011).

(38) Fredericks, S, Parrish, K, Sayre, D, Zhu, Q. PyXtal: A Python library for crystal structure generation and symmetry analysis. *Comp. Phys. Comm.* **261**, 0010-4655, 107810 (2021).

(39) Jain, A. et al. Commentary: The Materials Project: A materials genome approach to accelerating materials innovation. *APL Mater.* **1**, 011002 (2013).

(40) Curtarolo, S. et al. AFLOW: an automatic framework for high-throughput materials discovery. Comp. Mater. Sci. **58**, 218–226 (2012).

(41) Saal, J. E., Kirklin, S., Aykol, M., Meredig, B. & Wolverton, C. Materials design and discovery with high-throughput density functional theory: the open quantum materials database (OQMD). JOM **65**, 1501–1509 (2013).

(42) Yoo, A.B, Jette, M.A., & Grondona, M. SLURM: Simple Linux Utility for Resource Management. *JSSPP, Springer (Berlin),* **2862** (2003).

(43) Huang, B., Tang, Y. Research on optimization of real-time efficient storage algorithm in data information serialization. *PLOS ONE* **16**, 12, e0260697 (2021).

(44) Ong, S.P. et al. Monty: supplementary functions that are not part of the standard library. *Online GitHub Repository* https://github.com/materialsvirtuallab/monty (2022).

(45) The Cloudpickle collaboration. Cloudpickle: Extended pickling support for Python objects. *Online GitHub Repository* https://github.com/cloudpipe/cloudpickle (2022).

(46) The MessagePack organization. MessagePack: an extremely efficient object serialization library. It's like JSON, but very fast and small. *Online GitHub organization* https://github.com/msgpack (2022).

(47) The Redis collaboration. Redis: an in-memory database that persists on disk. *Online GitHub Repository* https://github.com/redis/redis (2022).

(48) Giantomassi, M. et al. AbiPy: Open-source library for analyzing the results produced by ABINIT. *Online GitHub repository* https://github.com/abinit/abipy (2022).

(49) Gonze, X. et al. ABINIT: First-principles approach to material and nanosystem properties. *Comp. Phys. Comm.* **180**, 12, 2582-2615 (2009).

(50) Mathew, K. et al. Atomate: a high-level interface to generate, execute, and analyze computational materials science workflows. *Comp. Mater. Sci.* **139**, 140–152 (2017).

(51) Kresse, G. & Furthmüller, J. Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Phys. Rev. B* **54**, 11169–11186 (1996).

(52) Thompson, A. P. et al. LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales, *Comp Phys Comm* **271,** 10817 (2022).

(53) Ganose, A. et al. JobFlow: a library for writing computational workflows. *Online GitHub repository* https://github.com/materialsproject/jobflow (2022).

(54) Jain, A. et al. FireWorks: a dynamic workflow system designed for high-throughput applications. *Concurrency Computat.: Pract. Exp.* **27**, 5037–5059 (2015).

(55) The Apache collaboration. Apache-AirFlow: A platform to programmatically author, schedule, and monitor workflows. *Online GitHub repository* https://github.com/apache/airflow (2022).

(56) The Prefect Collaboration. Prefect: the easiest way to coordinate your dataflow. *Online Github repository* https://github.com/PrefectHQ/prefect (2022).

(57) The Dask Collaboration. Dask: parallel computing with task scheduling. *Online Github repository* https://github.com/dask/dask (2022).

(58) Ong, S. P. et al. Custodian: A simple, robust and flexible just-in-time job management framework in Python. *Online Github repository* https://github.com/materialsproject/custodian (2022).

(59) Kresse, G.; Joubert, D. From ultrasoft pseudopotentials to the projector augmented wave method. *Phys. Rev. B* **59**, 1758 (1999).

(60) Perdew, J. P.; Burke, K.; Ernzerhof, M. Generalized Gradient Approximation Made Simple. *Phys. Rev. Lett.* **77**, 18, 3865–3868 (1996).

(61) Grimme, S.; Antony, J.; Ehrlich, S.; Krieg, H.A. consistent and accurate ab initio parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu. *J. Chem. Phys.* **132**, 154104 (2010).

(62) Anisimov, V. I., Zaanen, J. & Andersen, O. K. Band theory and Mott insulators: Hubbard U instead of Stoner I. *Phys. Rev. B* **44**, 943–954 (1991).

(63) Yu, M., Yang, S., Wu, C. et al. Machine learning the Hubbard U parameter in DFT+U using Bayesian optimization. *npj Comput Mater* **6**, 180 (2020).

(64) von Barth, U.; Hedin, L. A local exchange-correlation potential for the spin polarized case. *J. Phys. C: Solid State Phys.* **5**, 13, 1629–1642 (1972).

(65) Dasgupta, S., Lambros, E., Perdew, J.P. et al. Elevating density functional theory to chemical accuracy for water simulations through a density-corrected many-body formalism. *Nat Commun* **12**, 6359 (2021).

(66) Heyd, J. & Scuseria, G. E. Hybrid functionals based on a screened Coulomb potential. *J. Chem. Phys.* **118**, 8207–8215 (2003).

(67) Pulay, P. Convergence acceleration of iterative sequences. The case of SCF iteration. *Chem. Phys. Lett.* **73**, 2, 393-398 (1980).

(68) Henkelman G., Arnaldsson A., & Jónsson H. A fast and robust algorithm for Bader decomposition of charge density, *Comput. Mater. Sci.* **36**, 354-360 (2006).

(69) Dale, S.G.; Otero-de-la-Roza, A.; & Johnson, E.R. Density-functional description of electrides *Phys. Chem. Chem. Phys*. **16**, 14584-14593 (2014).

(70) Becke, A.D. & Edgecombe, K. E. A simple measure of electron localization in atomic and molecular systems. *J. Chem. Phys.* **92**, 9, 5397–5403 (1990).

(71) Walsh, A., Sokol, A.A., Buckeridge, J. et al. Oxidation states and ionicity. *Nature Mater* **17**, 958–964 (2018).

(72) Maintz, S.; Deringer, V.L.; Tchougreeff, A. L.; Dronskowski, R. LOBSTER: A tool to extract chemical bonding from plane-wave based DFT. *J. Comput. Chem.* **37**, 1030—1035 (2016).

(73) Jonsson, H.; Mills, G.; & Jacobsen, K.W. Classical and Quantum Dynamics in Condensed Phase Systems. *World Scientific* (1998).

(74) Trottier, R. M., Millican, S. L. & Musgrave, C. B. Modified single iteration synchronous-transit approach to bound diffusion barriers for solid-state reactions. *J. Chem. Theory Comput.* **16**, 5912–5922 (2020).

(75) Smidstrup, S., Pedersen, A., Stokbro, K. & Jónsson, H. Improved initial guess for minimum energy path calculations*. J. Chem. Phys.* **140**, 214106 (2014).

(76) Borlido, P., Schmidt, J., Huran, A.W. et al. Exchange-correlation functionals for band gaps of solids: benchmark, reparametrization and machine learning. *npj Comput Mater* **6**, 96 (2020).

(77) Wang, H., Zhang, L., Han, J., & Weinan, E. DeePMD-kit: A deep learning package for many-body potential energy representation and molecular dynamics. *Comp. Phys. Comm.* **228**, 178-184 (2018).

(78) Chang, J.H. et al. CLEASE: a versatile and user-friendly implementation of cluster expansion method. *J. Phys.: Cond. Matt.* **31**, 32 (2019).

CHAPTER 3: SIMMATE: A FRAMEWORK FOR MATERIALS SCIENCE

Adapted from: Sundberg, J.D. ; Benjamin, S.S. ; McRae, L.M. ; Warren, S.C. Journal of Open-Source Software, 7(75), 4364. Copyright 2022 Journal of Open-Source Software.

3.1 Introduction

Over the past decade, the automation of electronic structure codes has led to many large-scale initiatives for materials discovery, such as the Materials Project[1], AFLOW[2], OQMD[3], JARVIS[4], and others[5,6,7]. Each of these projects facilitates the creation and distribution of materials science data to the broader research community through databases, workflow libraries, and web interfaces. However, each software ecosystem (i.e., the collection of software used by a specific project) still possesses several pain-points for users attempting to implement new standards for computational research. No ecosystem provides a cohesive, vertical framework that runs without extensive user configuration. Proper setup involves learning how (i) workflows are defined/orchestrated, (ii) how databases are built/accessed, and (iii) how website interfaces/APIs make results accessible to the community, where each component requires learning a new package and, more importantly, learning how that package integrates with others. As a result, it can be difficult to integrate several smaller packages when building production-ready servers and databases for materials science research. To address this, we developed the Simulated Materials Ecosystem (Simmate).

3.2 Statement of Need

Simmate strives to simplify the process for researchers who are setting up a full-featured server. For the purposes of beginners, we desired a framework that could run locally without requiring any additional setup. For the purposes of experts, we sought to enable scaling of calculations across any number of resources and to facilitate the addition of new functionality. Simmate accomplishes these goals by (i) building on top of popular, well-established packages for workflow orchestration, database management, and materials science analysis and (ii) distributing our software as an "all-in-one" package that integrates all of these features. This is contrary to analogous software ecosystems that maintain custom packages and distribute solutions as separate programs.

For example, while other materials science ecosystems write workflow managers and task distribution from scratch (e.g., Fireworks[8] or AiiDA[6]), we instead use high-level, beginner-friendly packages such as Dask[9] and Prefect[10]. Using well-established packages also extends to our choice of website framework (Django[11]) and underlying materials analysis toolkit (PyMatGen[12]). The use of popular packages lets Simmate users take advantage of these packages' large user communities, abundant guides, and robust coding standards, while Simmate handles the integration of these packages in the context of materials science. This greatly facilitates the addition of new features while also enabling best practices.

To understand our aim to be an “all-in-one” solution, it is useful to compare Simmate with the collection of packages developed by the Materials Project[1]. The Materials Project is powered by many smaller packages, each with a specific use case – e.g., Atomate for a workflow library[13], Fireworks for workflow orchestration[8], Custodian for error handling[14], EMMET for schemas/APIs[15], MPContribs for third-party data[16], and many others. Each of these are powerful tools, but it can require significant effort and expertise to properly integrate several packages into a full-featured server. Meanwhile, Simmate contains modules for each of these features within a single, larger package – e.g., within our *workflows*, *workflow_engine*, *database*, and *website* modules. By maintaining these features within a single space, high-level integrated features can be more readily developed. This includes features that are unique to Simmate, such as dynamically built REST APIs and website interfaces. There is also a run-server command that compiles all features (including user-defined projects) with minimal setup. Many comparisons can be made between Simmate modules and existing packages from the materials science community, but most notably, Simmate focuses on the unification of components for high-level features and capabilities.

At the lowest level, Simmate is designed specifically for materials science research and the calculation of materials properties. While our current implementation is focused on periodic crystals and ab initio calculations, the framework is built around abstract data types and functionality. This allows easy integration of third-party software and databases. For example, we currently distribute data from other providers (COD[17], Materials Project[1], OQMD[3], and JARVIS[4]) as well as orchestrate calculations from popular DFT codes (e.g., VASP[18]). Each of these integrations benefits by inheriting from our core data types, which implement features such as error handling and job recovery for workflow integrations, as well as the automatic generation of Python APIs, REST APIs, and website interfaces for results and databases. Moreover, data can be converted to other useful Python objects, such as those from PyMatGen[12] or ASE[19], allowing further analysis of the materials.

Because Simmate removes many obstacles to advanced computation, we anticipate that this code will be utilized by beginners and experts alike. Thus, our tutorials are written for researchers who have never used the command-line or Python. However, as users become comfortable, they can begin exploring the underlying API and integrated packages for advanced features. Together, these features help Simmate bridge the gap between the existing ecosystems of materials science software while making production-ready implementations as easy as possible.

3.3 Supporting Information

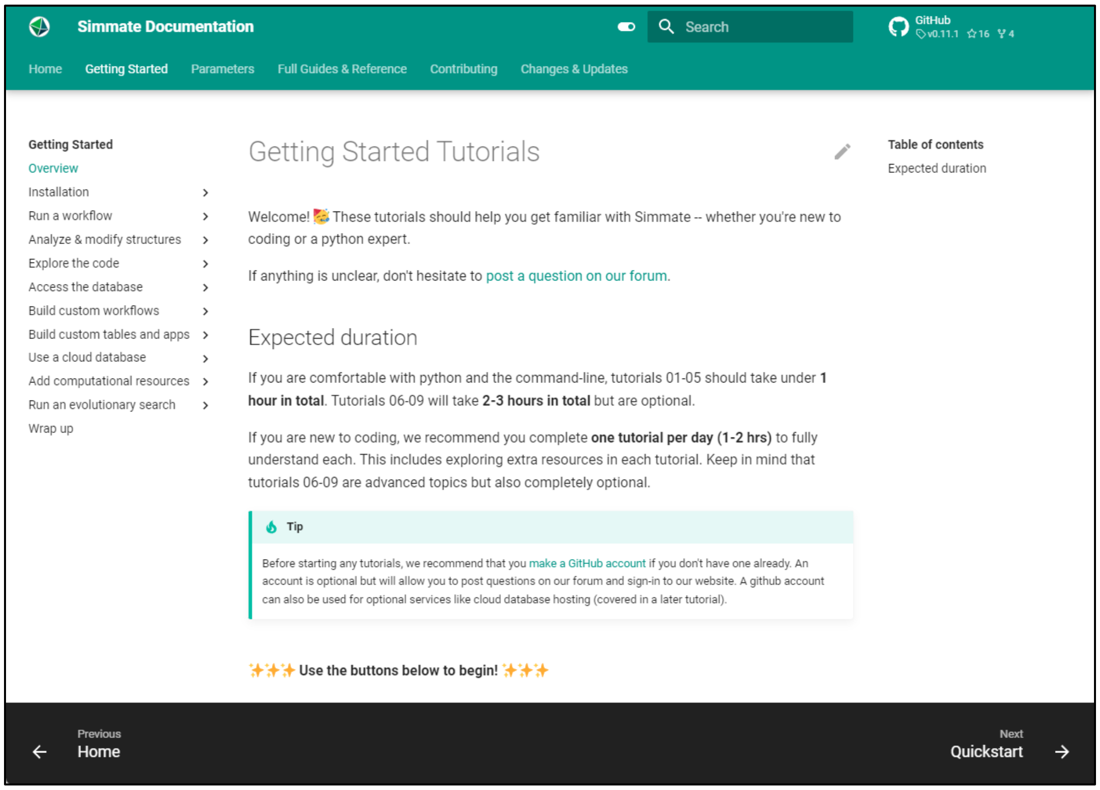
Herein, we provide additional context that was not included in the original 2022 JOSS manuscript. These supplementary sections give a brief overview of the functionality available in Simmate as well as how to explore our available resources. However, we emphasize that this is simply an overview of a few features and details on where more information can be found. The full specification and guides on how to use Simmate are contained within our source code repository, documentation, and website.

3.3.1 Documentation, guides, and tutorials

Exploration of Simmate's functionality is guided through our online documentation, which is currently hosted at https://jacksund.github.io/simmate/home/. The documentation includes a landing page, getting-started tutorials, full parameter specifications, in-depth API guides, and a change log (Figure 3-1). The many different categories serve to both guide beginners and enable advanced users to build out complex features.

Our guides have gone through many iterations of trials and feedback. This includes several lab workshops where >10 students work through tutorials and provide feedback/insight where necessary.

However, as Simmate continues to evolve, so will its documentation. There is always room for improvement. For this reason, all users may suggest changes to the documentation directly within the interface. Users may edit the documentation in plain text, and if changes are accepted, the documentation website is automatically rebuilt and redeployed through our GitHub CI.



Getting Started Tutorials

Welcome! 🥳 These tutorials should help you get familiar with Simmate -- whether you're new to coding or a python expert.

If anything is unclear, don't hesitate to post a question on our forum.

Expected duration

If you are comfortable with python and the command-line, tutorials 01-05 should take under **1 hour in total**. Tutorials 06-09 will take **2-3 hours in total** but are optional.

If you are new to coding, we recommend you complete **one tutorial per day (1-2 hrs)** to fully understand each. This includes exploring extra resources in each tutorial. Keep in mind that tutorials 06-09 are advanced topics but also completely optional.

Tip

Before starting any tutorials, we recommend that you make a GitHub account if you don't have one already. An account is optional but will allow you to post questions on our forum and sign-in to our website. A github account can also be used for optional services like cloud database hosting (covered in a later tutorial).

✨✨✨ **Use the buttons below to begin!** ✨✨✨

Figure 3-1. Simmate's online documentation. The guides are organized into searchable sections (white text, top of screen), where each section has many subcategories (black text, left column) to teach users how to use and add to the software.

3.3.2 A brief overview of modules

Because Simmate strives to be an all-in-one server and toolkit, there are many diverse features available. Although code that covers expansive features is typically broken down into several smaller software packages[8,12,13,14], we instead organize these features into submodules and apps. This allows each submodule and app to work out of the box and without configuration. At the time of writing, there are 12 top-level Simmate modules into which all features are organized (Table 3-1). Throughout our modules, we opt for verbose names. This allows beginners and experts to more rapidly navigate our code.

Many modules have direct analogs to those in other materials science packages, such as software of the Materials Project ecosystem[1] (Table 3-2). We therefore provide an "Alternatives" page that helps describe the relationships of Simmate features to other codes. Those familiar with materials science packages can easily find the direct replacement and, more importantly, recognize where they can apply familiar concepts within the much larger Simmate codebase. It is important to note that Simmate does not aim to be a total replacement of alternative software listed. Instead, our code illustrates how such features fit within a larger, more integrated ecosystem. This ultimately leads to significant changes in design that can guide future development of both Simmate and alternative programs.

Table 3-1. The top-level Simmate modules and their descriptions.

Module name	**Description**
calculators	third-party programs that run analyses for us (e.g., VASP or DeepMD)
command_line	makes some common functions available as commands in the terminal
configuration	the default Simmate settings and how to update them
database	defines how all Simmate data is organized into tables and lets you access it
file_converters	reformat to/from file types (e.g., POSCAR –> CIF)
toolkit	the fundamental functions and classes for Simmate (e.g., the Structure class)
utilities	contains simple functions that are used throughout the other modules
visualization	visualizing structures, 3D data, and simple plots
website	runs the simmate.org website and REST API
workflow_engine	tools and utilities that help submit calculations as well as handle errors
workflows	common analyses used in materials chemistry
conftest	utilities for running unit tests and only for contributing developers

Table 3-2. Comparison of Simmate to alternative materials science software.

Component	**Materials Project**	**Simmate**
Workflow submission	Fireworks[8]	Prefect, Dask, & *workflow_engine.execution* module
Workflow library	Atomate[13]	The *workflows* module and custom apps
Tasks & Error handling	Custodian[14]	The *workflow_engine* module
IO to different programs	pymatgen.io[12]	The *calculators* module
Database backend	MongoDB	SQLite, Postgres, or any engine supported by Django
Database API	EMMET[15]	The *database.base_data_types* module
Web API	mapidoc[1]	Built dynamically by the *website* module
Utilities & core classes	pymatgen[12]	The *toolkit* module (built w. pymatgen)
Website components	crystaltoolkit[20]	The *website.core_components* module
Third-party data	mpcontribs[16]	The *database.third_parties* module

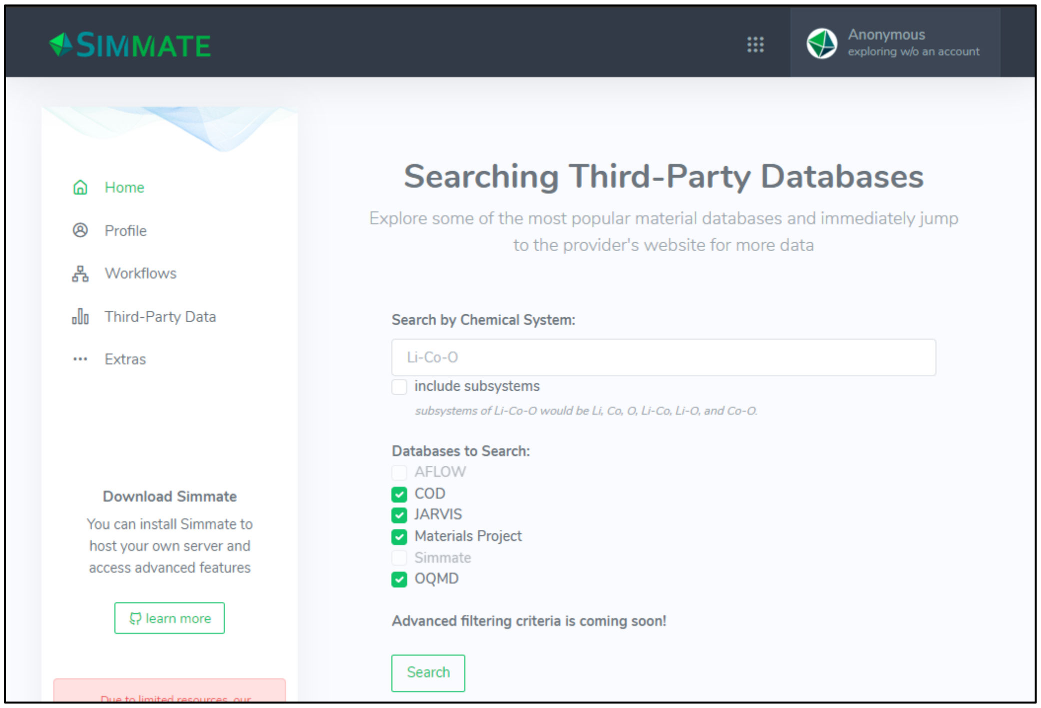
Note: many more comparisons can be made as well

3.3.3 A brief overview of user interfaces

There are several ways users can interact with Simmate's features – such as through our website, the command line, or Python. Each interface differs in its development status and what is required from the user.

First, new users (and especially those new to coding) will begin by interacting with the website interface at https://simmate.org/ (Figure 3-2). One can explore everything contained within the Simmate database, including user-run workflows as well as third-party data from COD, JARVIS, Materials Project, OQMD, and AFLOW. Users can also programmatically interact with our data using a standardized REST API interface (Figure 3-3). Unfortunately, however, even though the website is the first thing a user sees, it is often the last place functionality reaches due to limited time and development resources.

There are many more features available once Simmate is installed locally and then accessed through the command line interface (CLI) or Python. The CLI is reserved for the most commonly



Searching Third-Party Databases

Explore some of the most popular material databases and immediately jump to the provider's website for more data

Search by Chemical System: Li-Co-O

[ ] include subsystems

subsystems of Li-Co-O would be Li, Co, O, Li-Co, Li-O, and Co-O.

Databases to Search:

- [ ] AFLOW
- [✓] COD
- [✓] JARVIS
- [✓] Materials Project
- [ ] Simmate
- [✓] OQMD

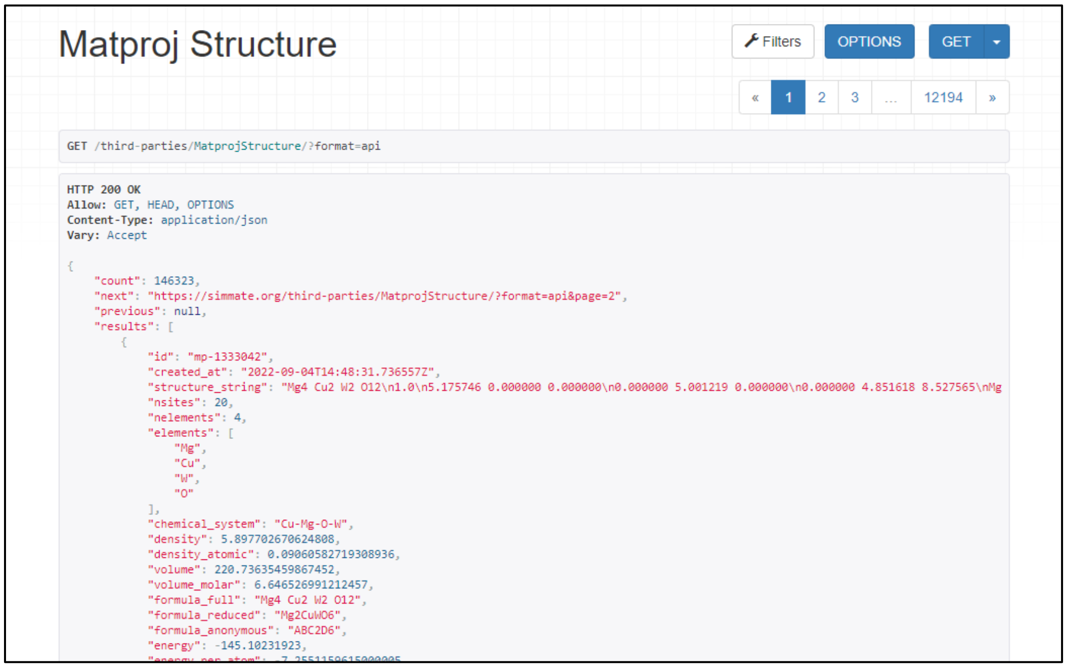
Advanced filtering criteria is coming soon!

Download Simmate

You can install Simmate to host your own server and access advanced features

Due to limited resources, our

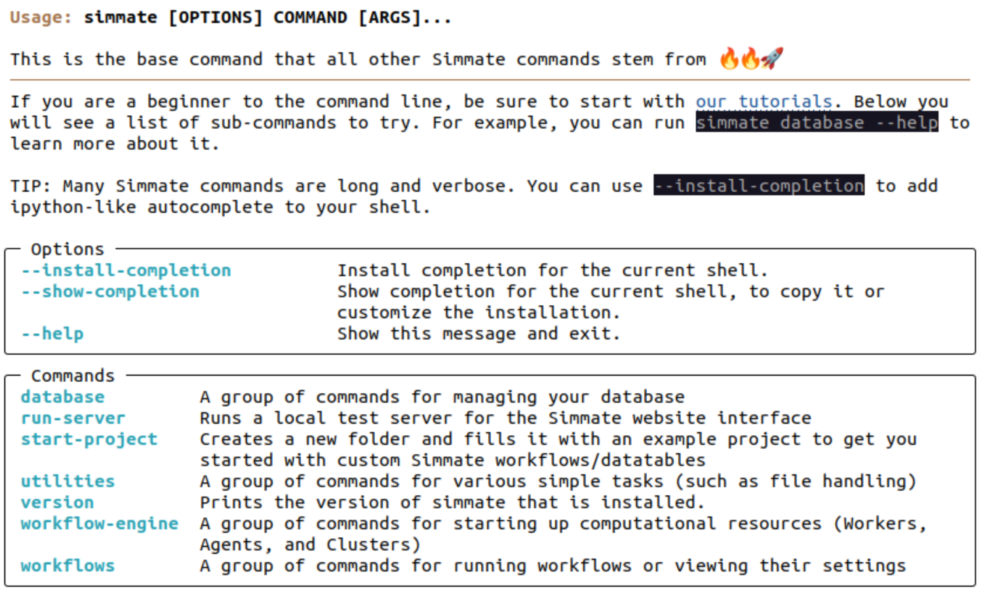
Figure 3-2. Simmate's website interface. This shows the homepage where users can search major databases. Advanced filtering criteria and additional API views are limited to the "Third-party Data" tab (left column).



```
GET /third-parties/MatprojStructure/?format=api
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "count": 146323,
    "next": "https://simmate.org/third-parties/MatprojStructure/?format=api&page=2",
    "previous": null,
    "results": [
        {
            "id": "mp-1333042",
            "created_at": "2022-09-04T14:48:31.736557Z",
            "structure_string": "Mg4 Cu2 W2 O12\n1.0\n5.175746 0.000000 0.000000\n0.000000 5.001219 0.000000\n0.000000 4.851618 8.527565\nMg
            "nsites": 20,
            "nelements": 4,
            "elements": [
                "Mg",
                "Cu",
                "W",
                "O"
            ],
            "chemical_system": "Cu-Mg-O-W",
            "density": 5.897702670624808,
            "density_atomic": 0.09060582719308936,
            "volume": 220.73635459867452,
            "volume_molar": 6.646526991212457,
            "formula_full": "Mg4 Cu2 W2 O12",
            "formula_reduced": "Mg2CuWO6",
            "formula_anonymous": "ABC2D6",
            "energy": -145.10231923,
            "energy_per_atom": -7.2551159615000005
```

Figure 3-3. Simmate's interactive REST API interface. The interface is built off the Django-REST framework, and one can quickly filter and paginate search results and directly convert them to JSON.



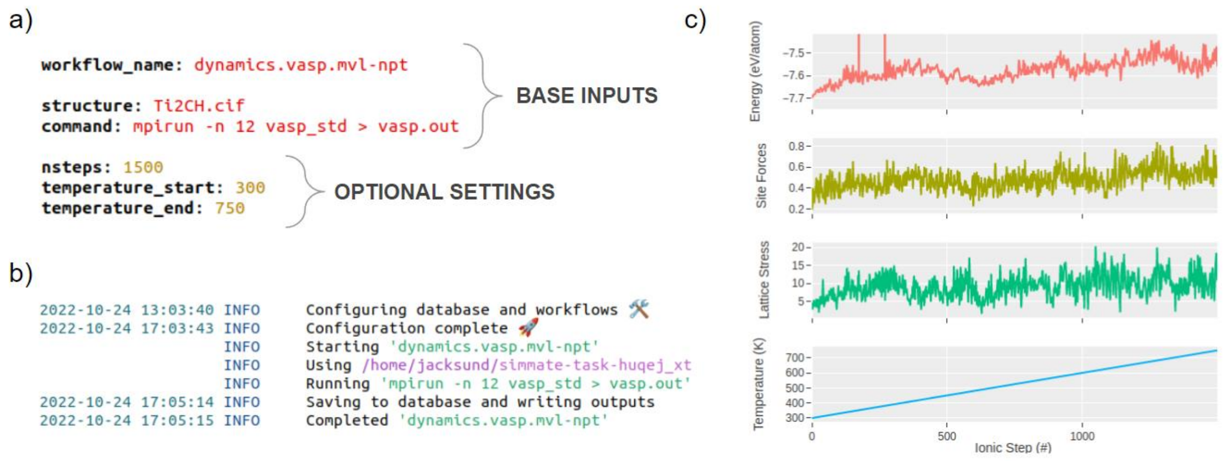
```
Usage: simmate [OPTIONS] COMMAND [ARGS]...

This is the base command that all other Simmate commands stem from 🔥🔥🚀

If you are a beginner to the command line, be sure to start with our tutorials. Below you
will see a list of sub-commands to try. For example, you can run simmate database --help to
learn more about it.

TIP: Many Simmate commands are long and verbose. You can use --install-completion to add
ipython-like autocomplete to your shell.

╭─ Options ──────────────────────────────────────────────────────────────────────────╮
│ --install-completion          Install completion for the current shell.            │
│ --show-completion             Show completion for the current shell, to copy it or │
│                               customize the installation.                          │
│ --help                        Show this message and exit.                          │
╰────────────────────────────────────────────────────────────────────────────────────╯
╭─ Commands ─────────────────────────────────────────────────────────────────────────╮
│ database         A group of commands for managing your database                    │
│ run-server       Runs a local test server for the Simmate website interface        │
│ start-project    Creates a new folder and fills it with an example project to get you│
│                  started with custom Simmate workflows/datatables                   │
│ utilities        A group of commands for various simple tasks (such as file handling)│
│ version          Prints the version of simmate that is installed.                  │
│ workflow-engine  A group of commands for starting up computational resources (Workers,│
│                  Agents, and Clusters)                                              │
│ workflows        A group of commands for running workflows or viewing their settings│
╰────────────────────────────────────────────────────────────────────────────────────╯
```

Figure 3-4. Simmate's command line interface. The output shown is from the base '*simmate --help*' command. Subcommands and options are listed below in teal-colored text.

accessed functions from each module. This includes utilities for building the initial database, submitting workflow runs, and connecting clusters/workers. The CLI attempts to closely match the Python module organization by grouping commands under similar names (Figure 3-3). Meanwhile, the Python interface interacts with the source code directly. All of Simmate's advanced features are therefore available in Python first, and the command line and website then follow for the most popular use cases.

3.3.4 Example workflow input and outputs

Users can easily build custom workflows but, most often, will begin with the prebuilt workflow library. Either way, workflows can be submitted through the website interface, command line, or Python. Currently, the command line is the preferred method. Submitting with the command line involves configuring a settings file and then running that file with Simmate, which handles the different stages of the workflow, organizing new files created, and loading results into the database. To illustrate, we will show two example workflow runs – one a single-step ab initio calculation and a second that orchestrates hundreds of ab initio calculations and analyzes the results.



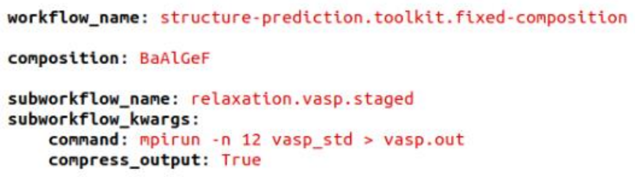
```
workflow_name: dynamics.vasp.mvl-npt

structure: Ti2CH.cif
command: mpirun -n 12 vasp_std > vasp.out

nsteps: 1500
temperature_start: 300
temperature_end: 750
```

```
2022-10-24 13:03:40 INFO     Configuring database and workflows 🛠
2022-10-24 17:03:43 INFO     Configuration complete 🚀
                    INFO     Starting 'dynamics.vasp.mvl-npt'
                    INFO     Using /home/jacksund/simmate-task-huqej_xt
                    INFO     Running 'mpirun -n 12 vasp_std > vasp.out'
2022-10-24 17:05:14 INFO     Saving to database and writing outputs
2022-10-24 17:05:15 INFO     Completed 'dynamics.vasp.mvl-npt'
```

Figure 3-7. Example workflow for a MD simulation. The (a) input YAML file, (b) monitoring and run log, and (c) example output plot are shown to help understand the different aspects of a workflow. For (c), plots are written as interactive HTML files using Plotly.

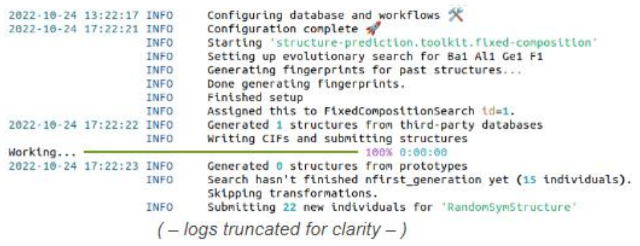
The first example is a single VASP calculation that executes a variable-temperature molecular dynamics simulation. One can generate a YAML (or TOML) settings file that describes what should be run (Figure 3-7a). At a minimum, only the workflow name and input structure need to be provided, but many optional parameters are also available to tune VASP parallelization, start/end temperatures, simulation time, structure standardization, and more. Given only these settings, calling the command ‘*simmate workflows run example.yaml*’ will gather input files (INCAR, POTCAR, POSCAR, etc.), run VASP, monitor (and fix) the job for common errors, read results into the database, and write summary files. For an MD run, the only summary file is a plot showing how energy, forces, and temperature vary during the simulation (Figure 3-7c).

a)



```
workflow_name: structure-prediction.toolkit.fixed-composition

composition: BaAlGeF

subworkflow_name: relaxation.vasp.staged
subworkflow_kwargs:
    command: mpirun -n 12 vasp_std > vasp.out
    compress_output: True
```

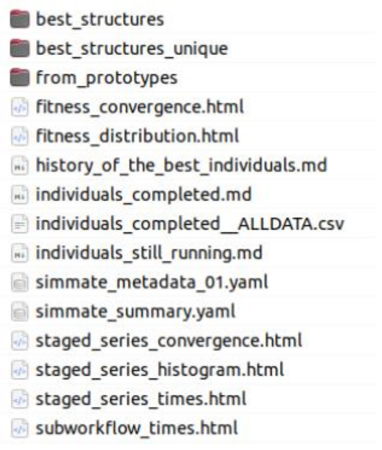
b)



```
2022-10-24 13:22:17 INFO     Configuring database and workflows 🛠️
2022-10-24 17:22:21 INFO     Configuration complete 🚀
                    INFO     Starting 'structure-prediction.toolkit.fixed-composition'
                    INFO     Setting up evolutionary search for Ba1 Al1 Ge1 F1
                    INFO     Generating fingerprints for past structures...
                    INFO     Done generating fingerprints.
                    INFO     Finished setup
                    INFO     Assigned this to FixedCompositionSearch id=1.
2022-10-24 17:22:22 INFO     Generated 1 structures from third-party databases
                    INFO     Writing CIFs and submitting structures
Working... ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00
2022-10-24 17:22:23 INFO     Generated 0 structures from prototypes
                    INFO     Search hasn't finished nfirst_generation yet (15 individuals).
                             Skipping transformations.
                    INFO     Submitting 22 new individuals for 'RandomSymStructure'
```

(– logs truncated for clarity –)

c)



```
best_structures
best_structures_unique
from_prototypes
fitness_convergence.html
fitness_distribution.html
history_of_the_best_individuals.md
individuals_completed.md
individuals_completed__ALLDATA.csv
individuals_still_running.md
simmate_metadata_01.yaml
simmate_summary.yaml
staged_series_convergence.html
staged_series_histogram.html
staged_series_times.html
subworkflow_times.html
```

d)

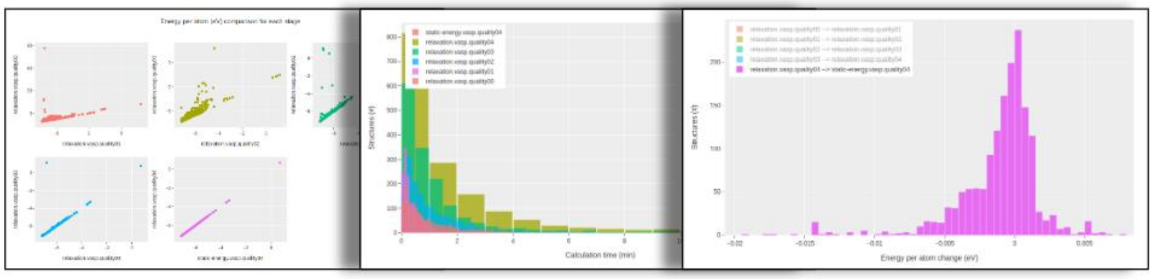


Figure 3-8. Example workflow for an evolutionary search. Screenshots are shown for (a) the input YAML file, (b) monitoring and run logs, (c) all output files and subdirectories, and (d) three of the interactive HTML plots available.

For the second example, we illustrate how Simmate can orchestrate hundreds of structure relaxations within an evolutionary search algorithm. This workflow generates many structures, runs a series of relaxations on each, mutates/strains the lowest energy structures, and analyzes all results on a cycle. Despite the many moving parts for a search, the configuration settings only require a composition.

There are, however, many optional settings that modify how structures are created or transformed, as well as how structure relaxation behaves (Figure 3-8a). Each individual relaxation has outputs analogous to the MD example, but for the higher-level evolutionary search, much more is available. The informatics framework makes it easy to evaluate search convergence, error in relaxation series, distinct structures, and much more (Figure 3-8c). With all results stored in the database, more in-depth analyses are possible as well.

REFERENCES

(1) Jain, A. et al. Commentary: The Materials Project: A materials genome approach to accelerating materials innovation. *APL Mater.* **1**, 011002 (2013).

(2) Curtarolo, S. et al. AFLOW: an automatic framework for high-throughput materials discovery. *Comp. Mater. Sci.* **58**, 218–226 (2012).

(3) Saal, J. E., Kirklin, S., Aykol, M., Meredig, B. & Wolverton, C. Materials design and discovery with high-throughput density functional theory: the open quantum materials database (OQMD). *JOM* **65**, 1501–1509 (2013).

(4) Choudhary, K., Garrity, K.F., Reid, A.C.E. et al. The joint automated repository for various integrated simulations (JARVIS) for data-driven materials design. *npj Comput Mater* **6**, 173 (2020).

(5) Drax, C. & Scheffler, M. The NOMAD laboratory: from data sharing to artificial intelligence. *J. Phys. Mater.* **2**, 036001 (2022).

(6) Pizzi, G., Cepellotti, A., Sabatini, R., Marzari, N. & Kozinsky, B. AiiDA: automated interactive infrastructure and database for computational science. *Comp. Mater. Sci.* **111**, 218–230 (2016).

(7) Talirz, L., Kumbhar, S., Passaro, E. et al. Materials Cloud, a platform for open computational science. *Sci Data* **7**, 299 (2020)

(8) Jain, A. et al. FireWorks: a dynamic workflow system designed for high-throughput applications. *Concurrency Computat.: Pract. Exp.* **27**, 5037–5059 (2015).

(9) The Dask Collaboration. Dask: parallel computing with task scheduling. *Online Github repository* https://github.com/dask/dask (2022).

(10) The Prefect Collaboration. Prefect: the easiest way to coordinate your dataflow. *Online Github repository* https://github.com/PrefectHQ/prefect (2022).

(11) The Django Collaboration. Django: The Web framework for perfectionists with deadlines. *Online Github repository* https://github.com/django/django (2022).

(12) Ong, S. P. et al. Python Materials Genomics (pymatgen): A robust, open-source python library for materials analysis. *Comp. Mater. Sci.* **68**, 314–319 (2013).

(13) Mathew, K. et al. Atomate: a high-level interface to generate, execute, and analyze computational materials science workflows. *Comp. Mater. Sci.* **139**, 140–152 (2017).

(14) Ong, S. P. et al. Custodian: A simple, robust and flexible just-in-time job management framework in Python. *Online Github repository* https://github.com/materialsproject/custodian (2022).

(15) The EMMET Collaboration. EMMET: be a master builder of databases of material properties. *Online Github repository* https://github.com/materialsproject/emmet (2022).

(16) The MPContribs collaboration. MPContribs: a platform for materials scientists to contribute and disseminate their materials data through Materials Project. *Online Github repository* https://github.com/materialsproject/MPContribs (2022)

(17) Gražulis, S. et al. Crystallography open database - an open-access collection of crystal structures. *J. Appl. Crystallogr*. **42**, 726–729 (2009).

(18) Kresse, G. & Furthmüller, J. Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Phys. Rev. B* **54**, 11169–11186 (1996).

(19) Larsen, A. H. et al. The atomic simulation environment—a Python library for working with atoms. *J. Phys.: Condens. Matter* **29**, 273002 (2017).

(20) The CrystalToolkit collaboration. CrystalToolkit: a framework for building web apps for materials science. *Online Github repository* https://github.com/materialsproject/crystaltoolkit (2022)

CHAPTER 4: HIGH-THROUGHPUT DISCOVERY OF FLUORIDE-ION CONDUCTORS VIA A DECOUPLED, DYNAMIC, AND ITERATIVE (DDI) FRAMEWORK

Adapted from: Sundberg, J.D.; Druffel, D.D.; McRae, L.M.; Lanetti, M.G.; Pawlik, J.T.; Warren, S.C. *npj Computational Materials*, **8**, 106 (2022). Copyright 2022 Nature Public Journals.

4.1 Summary

Fluoride–ion batteries are a promising alternative to lithium–ion batteries with higher theoretical capacities and working voltages, but they have experienced limited success due to the poor ionic conductivities of known electrolytes and electrodes. Here, we report a high-throughput computational screening of 9747 fluoride-containing materials in search of fluoride-ion conductors. Via a combination of empirical, lightweight DFT, and nudged elastic band (NEB) calculations, we identified >10 crystal systems with high fluoride mobility. We applied a search strategy where calculations are performed in any order (decoupled), computational resources are reassigned based on need (dynamic), and predictive models are repeatedly updated (iterative). Unlike hierarchical searches, our decoupled, dynamic, and iterative framework (DDI) began by calculating high-quality barrier heights for fluoride-ion mobility in a large and diverse group of materials. This high-quality dataset provided a benchmark against which a rapid calculation method could be refined. This accurate method was then used to measure the barrier heights for 6797 fluoride–ion pathways. The final dataset has allowed us to discover many fascinating, high-performance conductors and to derive the design rules that govern their performance. These materials will accelerate experimental research into fluoride–ion batteries, while the design rules will provide an improved foundation for understanding ionic conduction.

4.2 Introduction

Rapid progress in batteries that shuttle cations has led to their widespread use. Although lithium–ion batteries dominate the market for high-energy-density batteries, there are considerable efforts to develop alternate cation shuttles, including magnesium, sodium, and zinc[1]. Here, we explore a divergent strategy for ion-shuttle batteries: the development of materials that conduct fluoride (F^-)[2]. Comparatively, F^- has a low atomic mass, large theoretical capacity, and high natural abundance[3]. Furthermore, because it is the most electronegative element, F^- is stable against oxidation and could therefore yield batteries that operate at extreme voltages.

Despite these promising properties, fluoride-ion batteries (FIBs) have received little development. This is largely because the few existing fluoride-ion conductors have modest transport and low stability[4,5,6]. For example, the first reversible FIB, reported in 2011, achieved good fluoride-ion mobility only at 150 °C[7]. Since this initial breakthrough, improved conductors have been considered for FIBs, including $BaSnF_4$, $PbSnF_4$, $Sm_{0.95}Ca_{0.05}F_{2.95}$, and $Ce_{0.975}Sr_{0.025}F_{2.975}$, which all achieve improved conductivity ($>1 \times 10^{-4}$ S cm^{-1} at 20 °C; ~300 meV energy barrier) but lack stability at extreme potentials[8,9,10,11,12,13]. This challenge with stability was especially notable with our development of Y_2CF_2 and Sc_2CF_2 as anodes, which operate at extreme potentials that fall outside the stability window of most fluoride–ion electrolytes[3,14,15,16]. Therefore, there is a strong need to identify fluoride-ion conductors.

Because so few materials have been examined as possible fluoride-ion conductors, we suspected that there may be many high-performance materials that are yet undiscovered. While most studies on mobility have surveyed a limited number of structure types or systems[17,18,19,20,21,22,23,24,25,26], large-scale analyses of ionic mobility[27,28,29,30] have used hierarchical calculations (Figure 4-1a). In the hierarchical approach, simplified models are used to select candidates for more rigorous calculations, such as nudged elastic band (NEB) or molecular dynamics (MD). This is an efficient approach for generating ionic mobility databases and identifying promising materials, but it also has limitations. Most importantly, the existence of selection criteria requires a good understanding of the materials being studied, which was not the case for F^-conduction. In addition, the removal of candidates at early stages does not allow one to assess whether the selection criteria are biased or incorrect; final candidates simply reinforce current models. These limitations of standard hierarchical studies led us to examine an alternate approach in our search for fluoride-ion conductors.

To this end, we employ a high-throughput search of fluoride-ion conductors using a decoupled, dynamic, and iterative (DDI) framework (Figure 4-1b). In this framework, calculation stages are not hierarchical but are instead performed in any order (decoupled), predictive models are repeatedly updated during the search (iterative), and computational resources are reassigned as models change and materials are re-ranked (dynamic). A full account of the DDI framework will be presented elsewhere; instead, this manuscript describes the application of the DDI framework to the search for fluoride-ion conductors. In this search, we screened approximately 10,000 fluoride-containing structures, identified several crystal systems as promising conductors, and developed heuristics for quickly assessing ion transport in other materials. These materials will inspire future investigations of electrodes and electrolytes, while the heuristics will enable future high-throughput studies and provide a basic understanding of the factors that govern fluoride transport.

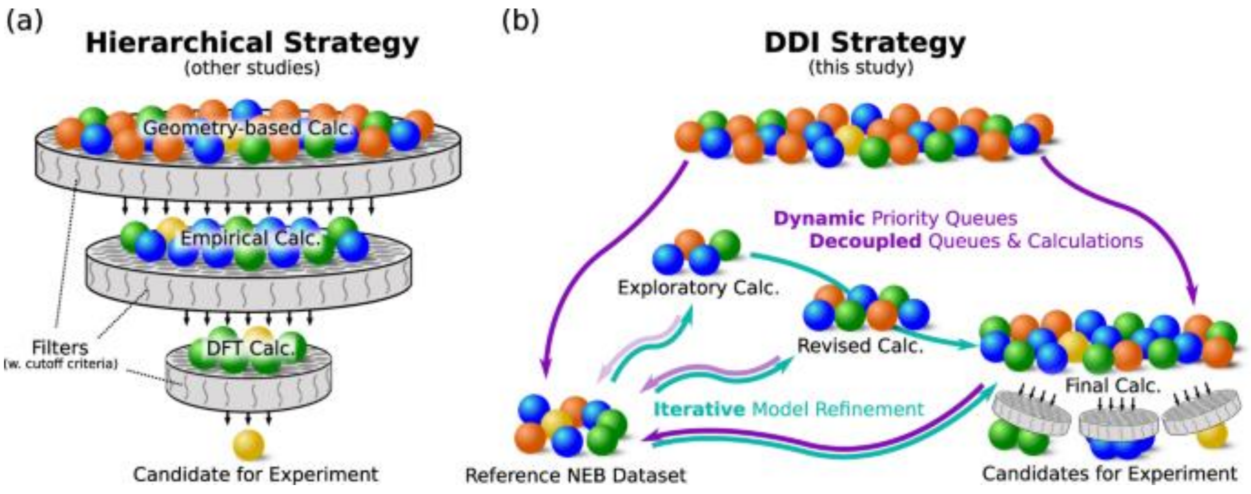


Figure 4-1. Comparison of hierarchical and DDI (dynamic, decoupled, and iterative) search strategies. (a) Hierarchical searches perform successive calculations of increasing quality and computational cost, but low-quality predictors are used to remove most candidates. This strategy is possible only when heuristics are already known, and it does not allow selection criteria to change at later stages. (b) The DDI strategy used in this study creates a high-quality reference dataset, which is then used to create and iteratively refine rapid calculations. We used this approach to develop a rapid calculation for fluoride mobility and to predict barrier heights for a large pool of candidates.

4.3 Results

4.3.1 Design strategy of the DDI framework

The typical strategy for high-throughput searches is to proceed from low-quality to high-quality calculations in distinct stages and with fixed cut-off criteria. However, when no preexisting models or heuristics exist, early-stage criteria cannot be accurately selected, and good candidates may be erroneously removed due to the larger error involved in the initial calculations. Moreover, the removal of candidates without subjecting them to high-quality calculations does not allow one to test the accuracy of the selection criteria. We, therefore, developed a framework where all workflows can be run independently, and low-quality calculations are not required prerequisites for intensive calculations. This allowed us to produce a constantly growing set of high-quality NEB calculations against which the low-quality calculations could be iteratively assessed. These assessments allowed us to refine our models and reallocate computational resources as the high-throughput search progressed. This strategy can be described as decoupled, dynamic, and iterative (DDI).

The DDI strategy is possible with workflow management software commonly used in the materials science community (such as AiiDA[31], Atomate[32,33], AFLOW[34], and others[35,36,37,38]), but we utilize a custom framework in this study. While a technical discussion of the framework will be presented elsewhere, our source code has been made publicly available[39], and we provide a high-level comparison to previous search strategies in the supporting information. Overall, computational resources are allocated via a centralized database server and asynchronous task executors. We define a task executor as a computational resource that requests and runs the candidates based on a priority ranking. For example, if the priority function of an NEB calculation is based solely on pathway length, the task executor will search the database for the shortest pathway yet to be calculated and run the NEB calculation. Task executors work through the database, and candidates may be deprioritized but are never removed. Thus, the DDI strategy creates an ongoing queue where uncompleted calculations are ranked according to priority.

4.3.2 Generating a pool of candidate pathways

The DDI framework requires a database of candidates, where dynamic priority ranking allows us to add new candidates even after the search has started. For our fluoride mobility calculations, individual candidates are not structures but instead pathways within a structure. This is because structures can contain multiple symmetrically unique diffusion pathways, where each unique pathway should be evaluated independently. A crystal's cumulative diffusion network and long-range barrier for transport can then be evaluated using dimensional analysis, as described in our supporting information. Thus, we generated a starting pool of candidate pathways, built into the option to introduce new pathways, and identified promising candidates using the lowest-energy percolating pathway.

We used all fluoride-containing structures from the Materials Project database (version 2021-02-08)[40], yielding 9747 structures. Symmetrically unique pathways – each represented by a unique combination of start and end crystallographic sites – were identified for each structure using pymatgen-diffusion's distinct path finder[41] with two criteria: (1) pathways were limited to 5 Å in length, and (2) only the five shortest symmetrically unique pathways per structure were considered. Although these choices limited our collection of candidate pathways, we examined whether these longer pathways were physically reasonable. Individual pathways longer than 5 Å required fluoride movement beyond its first- or second-nearest neighbor, leading to unfavorable interactions (Supplementary Figure 4-S3), and we also found that the shortest pathways within a structure yielded the lowest barriers (Supplementary Figure 4-S4). Thus, we concluded that loosening these criteria (leading to the addition of new candidate pathways) was unnecessary in the scope of this study. This produced 43,352 candidate pathways, and no other criteria were applied to limit or remove candidate pathways. These 40,000+ pathways establish a constant work-in-progress for the most computationally intensive calculations such as NEB, and we report on that progress herein.

4.3.3 Generating a high-accuracy reference dataset

The first step in our evaluation of candidates was the calculation of a high-quality reference dataset. Compared to standard hierarchical workflows, which perform high-quality calculations only on candidates selected by low-quality predictions, we instead selected a diverse subset of materials for these high-quality calculations. This provided reference values against which we could judge and iteratively refine our low-quality predictions. The DDI strategy allowed this dataset to cumulatively grow during our study (currently, N = 299), which further enabled iterative evaluation of low-quality predictions.

The most rigorous calculation used to assess fluoride transport was a midpoint-only NEB relaxation. Specifically, we calculated the diffusion barrier for pathways by fully relaxing the start, end, and midpoint supercell structures (R = 10 Å) via NEB, where the midpoint structure was prerelaxed by an image-dependent pair potential surface (IDPP) to improve our starting point[42]. Even though full relaxations were performed, the use of a single structure in NEB means that the predicted barrier is only the minimum possible barrier for the pathway. This is because the true transition state may lie off the midpoint, leading to a higher barrier than what is calculated here. Thus, this calculation can be viewed as an extension of the Trottier et al.[43] explicit-error approach combined with empirical prerelaxations utilized by Smidstrup et al.[42]. This was selected over a full climbing image NEB[44] analysis of promising pathways because the supercell relaxations required are extremely expensive, even with substantial efforts[45,46,47,48] made in this area.

In building this reference dataset, various priority-ranking functions were utilized by our task executors, and these functions evolved throughout our search. Initial calculations were orchestrated by random selection of pathways. The reference dataset was then used to identify trends and establish heuristics, and these models prioritized future calculations for the reference dataset. Thus, as our reference dataset grew, we updated our models and their influence on priority queues (Figure 4-1b). The flexibility of our priority functions also allowed us to focus on specific structure types or compositions in a subset of executors. This was particularly useful when we identified an interesting structure type and wanted to analyze similar candidates by NEB. Together, all these executors and various ranking functions worked to cumulatively produce the final dataset of NEB results.

At the time of publication, 299 pathways successfully converged for this midpoint-only NEB calculation, and all these pathways are listed in Supplementary Table 4-S1. These reference calculations allowed us to evaluate the lower quality calculations described below.

4.3.4 Developing a rapid calculation within the DDI framework

Because the computational cost of NEB limited its use to fewer than 300 pathways out of >40,000 (<0.7% total), we sought to identify a faster calculation to estimate energy barriers. We therefore used small supercells ($R = 7$ Å) on the initial, final, and midpoint structures to calculate energy barriers, but these smaller supercells are still similar in size to those used in other high-throughput searches[27,28,29,30]. To further reduce the computational cost, the initial and final structures were not relaxed, while the midpoint image was relaxed using the IDPP method[42]. IDPP is an empirical method that uses bond lengths to produce a relaxed structure that is similar to that produced by NEB. With these steps to reduce computational cost, it became feasible to calculate thousands of pathways rather than hundreds. At present, we have calculated 8497 pathways via this method, and 6797 (80%) have been completed successfully.

We expected that these faster calculations would have a large error because of electrostatic repulsion among small supercells[49], the absence of complete relaxation, and the possibility that the transition state was not located at the pathway's midpoint[43]. To quantify the error in these calculations, we compared them to the more rigorous midpoint-only NEB calculations described above. We observed a moderate correlation between the approximated barriers (E_{approx}) and the NEB barriers (E_{NEB}) (Fig. 4-2, green), with a median error (E_{approx}-E_{NEB}) of +165 meV and a standard deviation of 427 meV. This is a large error distribution compared to typical DFT barrier predictions. Nevertheless, the approximation successfully isolates pathways with low-energy barriers. Specifically, 165 out of 167 (98.8%) pathways predicted to have $E_{approx} < 1.0$ eV, in fact, have $E_{NEB} < 1.0$ eV. Despite this agreement, our approximation also results in a large number of false negatives, where 6 out of 29 (20.7%) pathways with $E_{approx} > 1.5$ eV actually have $E_{NEB} < 1.0$ eV. The tendency to overestimate the barrier is consistent with the use of a small supercell, which can destabilize the transition state[49]. However, because this error is systematic, it can be corrected.

To reduce the systematic error in E_{approx}, we used a linear regression between our static approximation and fully relaxed NEB. The linear regression was updated as we collected NEB data, which allowed us to iteratively improve our empirical correction. At the time of publication, the empirical correction (in eV) is:

$$E_{NEB} = 0.549 \times E_{approx} + 0.119$$

This correction greatly reduces the systematic error (Figure 4-2, blue). The median error is reduced from +165 meV to −9 meV, and the standard deviation is reduced from 427 meV to 318 meV.

With the goal of further improving the approximation, we explored additional strategies to refine the trade-off between computational cost and error reduction. The DDI framework allowed us to explore alternative versions of the approximated calculation because we can initiate task executors that are independent of preexisting executors. We therefore examined how alternative relaxation convergence criteria would affect the calculation's efficiency and accuracy. Start, midpoint (NEB), and end supercell structures (R = 7 Å) were relaxed to an ionic convergence of 0.5 meV, and all ionic steps were analyzed for convergence and CPU usage (Supplementary Figure 4-S5). This allowed us to understand how convergence criteria affect computational cost and calculation accuracy.

Our results show that as the convergence criteria are improved, the calculation accuracy also improves. However, decreasing the convergence criteria below 100 meV does not further improve the errors' standard deviation, likely because the accuracy is not limited by convergence but rather by the use of a small supercell. Compared to a static-IDPP calculation, stopping the calculation at a 100 meV convergence increased the computation cost by 1.6x, decreased the median error from 165 to 92 meV, and reduced the standard deviation in the error from 427 to 330 meV (Fig. 2, red). We sought to further reduce the error using an empirical correction of the data, as we did above. We performed a linear regression on our data but added a second term that sought to account for the residual forces in the structure due to incomplete relaxation:

$$E_{NEB} = 0.856 \times E_{approx} - 0.153 \times F_{rel} + 0.033$$

where F_{rel} is a measure of the forces that remain on atoms in the incompletely relaxed structures. F_{rel} is found by calculating the Euclidean vector norm of the force for all atomic sites in each of the start, end, and midpoint structures; F_{rel} is the difference in the Euclidean vector norm between the midpoint and start (or end) structures, whichever is larger.

The F_{rel} empirical coefficient of −0.153 implies that structures with higher F_{rel} will often overestimate barriers. Although F_{rel} has the primary purpose of correcting for incomplete relaxation, it also provides a partial correction for our use of small supercells[49].

When linear regression is applied as an empirical correction, this fit yields an updated median error of −10 meV and a standard deviation of 286 meV. This is a drastic improvement from our original uncorrected static calculation that gave a median error of 165 and standard deviation of 427 meV. However, this is only a modest improvement in accuracy compared to the empirically corrected static calculation, which had a median error of −9 meV and standard deviation of 318 meV. Given that this small improvement in accuracy comes with a 60% increase in computational cost, we decided that our empirically corrected static approximation (described in the preceding section) would best balance computational costs with accuracy. Thus, by using this approximation, we were able to calculate 6797 barriers to identify promising materials. This overall strategy was enabled by our DDI framework, which simplified the process of testing different approximations and iteratively correcting errors without interrupting or restarting the search.

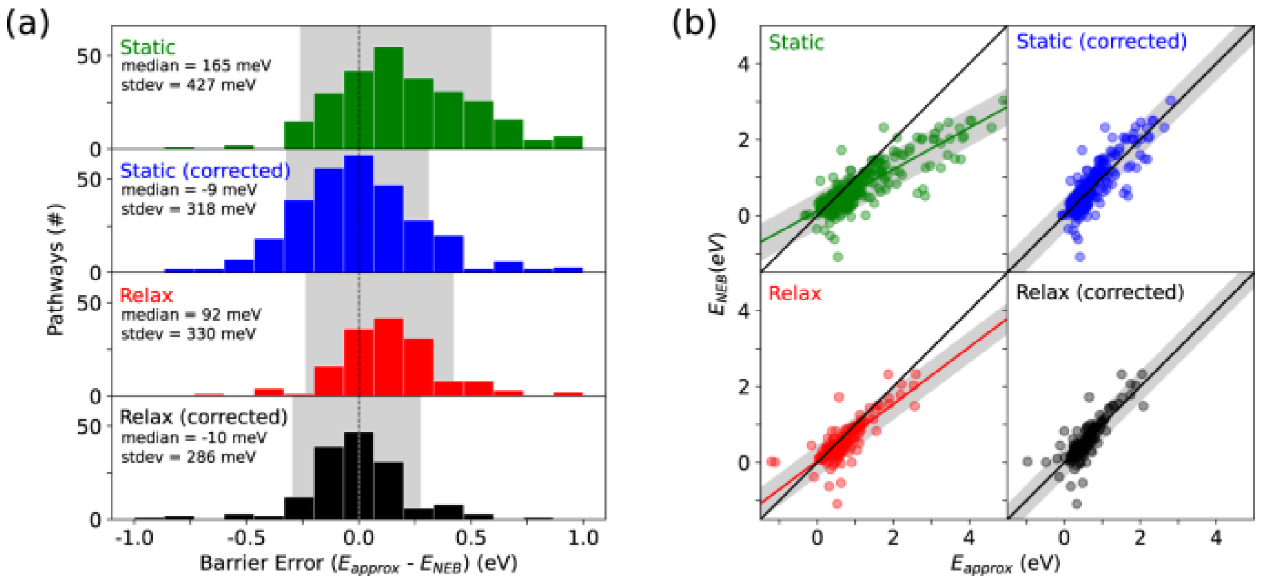


Figure 4-2. Error distributions of approximated barrier heights relative to reference NEB barrier heights (R = 10 Å). The static pathway (green) as well as the partially relaxed pathway (red) used smaller supercells (R = 7 Å). Empirical corrections to barrier height were made for the static (blue) and partially relaxed (black) calculations using linear regression as described in the text. The partially relaxed calculations (red, black) were performed on a subset of pathways (N = 170). Gray regions are centered about the mean in (a) and the linear regression in (b), where the region width shows the standard deviation.

4.3.5 Identifying heuristics for fluoride-ion transport

At the time of publication, we completed 6797 empirically corrected static calculations. These calculations provide a high-quality dataset for fluoride-ion transport and provide us with the opportunity to identify heuristics. In fact, during the course of this study, we iteratively assessed the growing database for trends. This allowed us to use emerging trends to prioritize calculations on materials with promising characteristics.

Although heuristics for fluoride-ion transport have not been described previously, heuristics for cation transport have been developed[30,45,50,51,52]. For example, the size of free-volume networks, bond lengths, and pathway lengths were used to predict the ionic mobility of lithium, sodium, and zinc[27,28,29,30]. These heuristics were also applied to empirically predict fluoride transport in 1,500 materials[53]. Because of our recent discovery of a distinct transport mechanism in some anion conductors[3], it is unlikely that heuristics for cation transport generally apply to anions. In the following section, we sought to use our ab initio calculations to evaluate whether heuristics based on sterics, charge, and pathway length could be applied to fluoride diffusion.

We evaluated the sterics of the diffusing atom by treating atoms as hard spheres and measuring their overlap along the diffusion pathway. The overlap was quantified using the change in ionic radii overlap (ΔIRO), where ionic radii were obtained by using bond valence analysis. The ΔIRO descriptor uses the same underlying principles of sterics and charge as other software, such as SPSE[27] and SoftBV[50], which evaluate free volumes for cationic diffusion.

Initially, we hypothesized that ΔIRO would correlate strongly with barrier height because overlap between fluoride and neighboring atoms in the transition state would be unfavorable. Unexpectedly, we did not find a clear relationship between ΔIRO and barrier height. As we analyzed our data, however, we found that materials with a small ΔIRO and high barrier often had pathways where significant bond-breaking occurred. We also observed that increasing overlap between fluoride and anions (cations) resulted in larger (smaller) barriers. We therefore refined our model to incorporate both increasing overlap (positive ΔIRO) and bond breaking (negative ΔIRO) and distinguished between neighbors that were cations (ΔIRO_{cation}) and anions (ΔIRO_{anion}). For each type of ΔIRO, we used the maximum value for F^- with each of its neighbors. The revision of our incorrect hypothesis highlights the value of the DDI strategy in contrast to traditional hierarchical searches. If this had been a hierarchical search, our incorrect hypothesis would have removed promising materials, and we might not have identified our error.

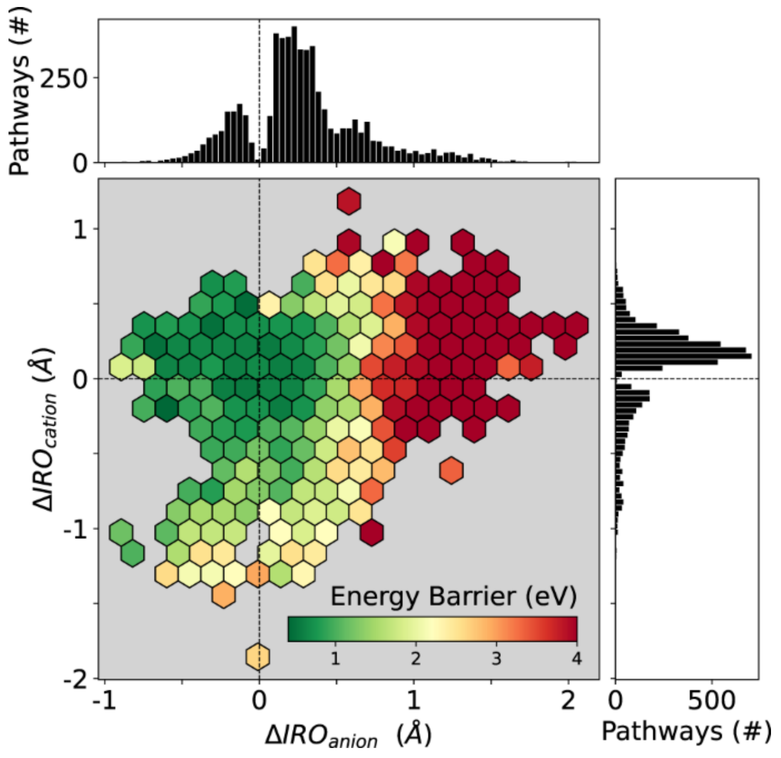


Figure 4-3. Understanding the role of sterics and charge on barrier height based on 6797 pathways. The diffusing fluoride's change in ionic radii overlap was separately evaluated for neighboring anions (ΔIRO_{anion}) and cations (ΔIRO_{cation}). All measurements were performed on the IDPP-relaxed path in R = 7 Å supercells, and the barrier corresponds to the static, midpoint-only approximation discussed in the main text. Each hexagon bin is colored using the average barrier of all pathways in that region.

This updated model is presented in Figure 4-3. The smallest barriers for fluoride transport occur when, in the transition state, the fluoride moves towards cations (positive ΔIRO_{cation}) and away from anions (negative ΔIRO_{anion}). Small barriers can also occur with a modest decrease in fluoride-anion distances or a modest increase in fluoride-cation distances. However, when fluoride-anion distances decrease more than ~0.5 Å or when fluoride-cation distances increase more than ~1.0 Å, barriers rapidly grow. Overall, the data show that anions in the vicinity of the diffusing fluoride have an especially profound influence on fluoride transport.

Even though this updated model considers the distinct roles of cations and anions, it does not consider the number of ions or the magnitude of their charge. To measure the change in electrostatic energy of the diffusing ion, we calculated the Ewald energy along the empirically relaxed pathway

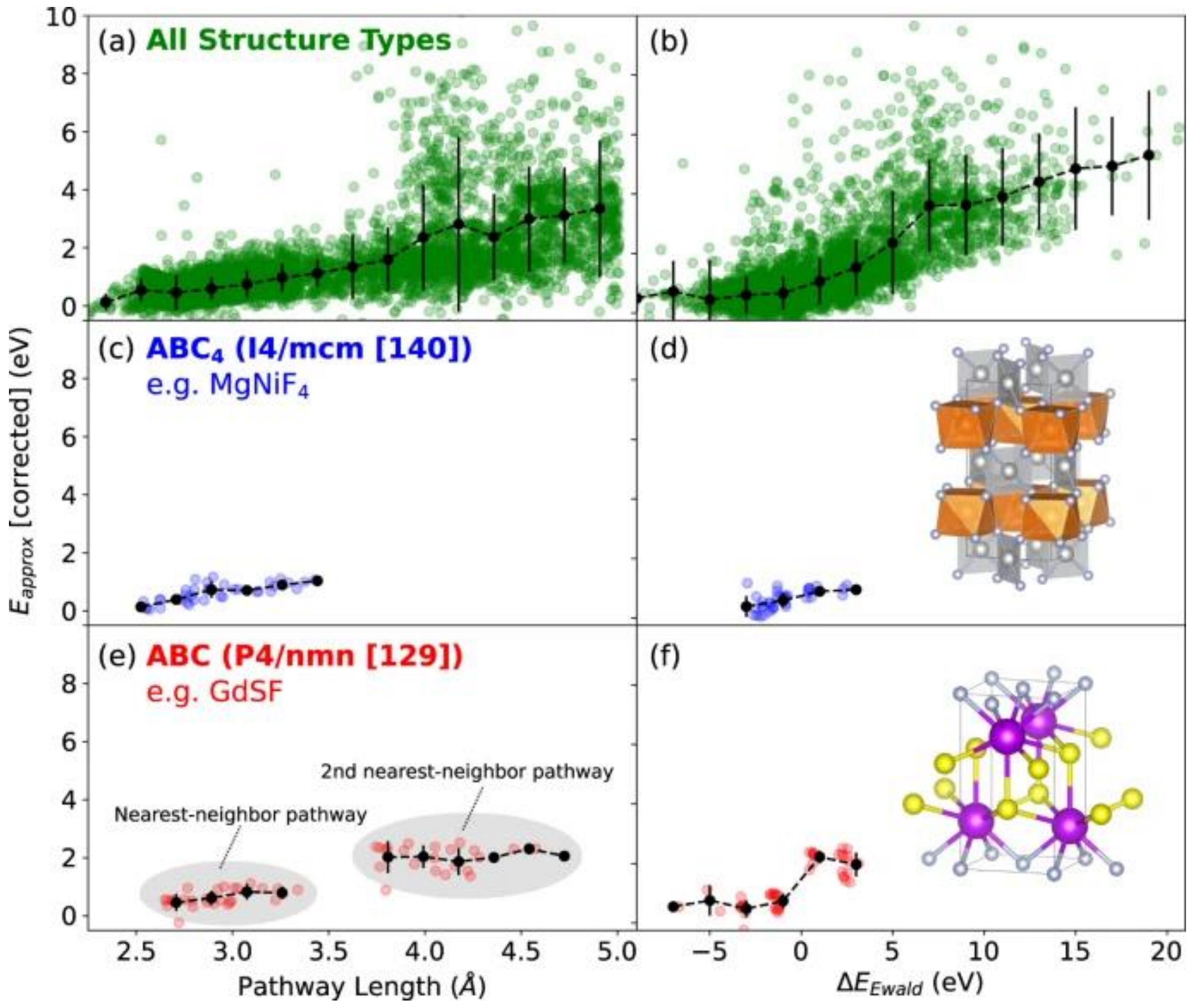


Figure 4-4. Understanding fluoride transport using pathway length (left column, a, c, e) and the change in Ewald energy (ΔE_{Ewald}) (right column, b, d, f). Pathways for all structure types show moderate correlation with (a) pathway length and (b) ΔE_{Ewald}. When pathways were analyzed structure type-by-structure type (c–f), we observed much tighter distributions for barrier heights. However, many structure types display barrier heights that are independent of these predictors (Supplementary Figure 4-S6). Trendlines and error bars for each subplot depict the mean and standard deviation of barrier heights within binned regions, respectively.

(ΔE_{Ewald}). Here, we expected that positive changes in Ewald energy (corresponding to more unfavorable electrostatic environments) would occur in higher energy pathways. This is corroborated by the positive trend of ΔE_{Ewald} with approximated DFT-calculated barriers (Figure 4-4b). Nearly all pathways with an unfavorable change in ΔE_{Ewald} greater than 5 eV have a barrier greater than 1.5 eV, which indicates that ΔE_{Ewald} is a promising empirical predictor of high barrier pathways.

The final heuristic that we present is based on the hypothesis that longer pathways have higher barriers[54]. In Figure 4-4a, we plot pathway length and barrier height and find that virtually all pathways shorter than 3.8 Å have barriers under 2.0 eV, but pathways longer than 3.8 Å often have much larger barriers. The radial distribution function in Supplementary Figure 4-S3 helps explain these observations: pathway lengths over 3.8 Å often require that fluoride moves not to a nearest-neighbor fluoride vacancy but to a second nearest-neighbor fluoride vacancy. Therefore, fluoride-ion conductors generally have lower barriers for shorter paths (Supplementary Figure 4-S4), which is consistent with heuristics for cation diffusion.

Up to this point, this section shows how heuristics can be applied to the entire population of fluoride-ion conductors. However, the large standard deviation for each of these relationships shows that these heuristics have limited explanatory or predictive value. We therefore sought to understand whether the standard deviation could be reduced by comparing only those structures that share the same structure type. Several exemplar structure types are presented in Figure 4-3 and the SI (Supplementary Figure 4-S6). Interestingly, we find that these broad heuristics (e.g., barrier height increases with pathway length) describe a small number of structure types but that these heuristics are incorrect for most structure types. For example, we observe that the barrier height remains constant as the pathway length increases in many structure types. It therefore becomes clear that our general heuristics are often untrue for specific structure types, even if they are true for the overall population.

Although we find that general heuristics are poor predictors of ionic mobility, in specific structure types, our plots of barrier height vs. pathway length or ΔE_{Ewald} have very small standard deviations. These small standard deviations suggest that, when analyzed structure type-by-structure type, pathway length and Ewald energy are actually good predictors of barrier height. For example, given a specific structure type and pathway length, our dataset allows us to accurately predict the barrier height of a yet-unknown material. The fact that simple structural or electrostatic features can yield the barriers of unexplored compositions will be extremely useful in future explorations of other materials. Furthermore, we propose that our final dataset can be used to identify features beyond those found in this study. This can also extend to the development of complex descriptors via machine-learning and active-learning approaches, which may outperform the predictive capabilities of general heuristics.

4.3.6 Exploring the final dataset

By applying the DDI strategy to a large group of fluoride-containing crystals, we calculated 6797 approximate barriers and 299 NEB barriers. Because of the iterative refinement, the error in the approximate barriers is modest (std. deviation = 318 meV). While the magnitude of this error indicates that E_{approx} should not be compared directly with experimental values, our results are sufficiently accurate to identify promising materials for further study. Moreover, although our calculations focused on single-vacancy diffusion in single crystals, many candidates may benefit from exploration at different stoichiometries and higher vacancy concentrations. To encourage follow-up studies on these materials, the full dataset is made available with common querying flags (e.g., hull energy) in the SI. To illustrate this dataset's contents, we outline and apply three example use cases: (a) exploration of specific compositions, (b) identification of structure types for 2-D fluoride intercalation, and (c) identification of inexpensive solid-state electrolytes.

Table 4-1 Subset of promising Pb-containing phases. The calculated band gaps aid in identifying applications.

Material	**Materials project ID**	**E_{approx} (meV) (all paths listed)**	**Band gap (eV)40**
PbF_4	mp-341	497, 724, 770, 1057	2.00
PbF_2	mp-315	542, 1141	4.44
$SnPbF_4$	mp-20815	323, 467, 610, 779	3.46
$BiPb_2F_7$	mp-1227492	166, 367, 387	2.66
Pb_3IF_5	mp-1220040	182, 278	3.12
$AgPbF_6$	mp-1206101	232, 724	0.22
$BaPbF_6$	mp-19799	419, 495, 645, 916	2.89
$PdPb_2F_6$	mp-1209528	157, 160, 371	1.13

First, we sought to identify conductors in our dataset based on composition, and because our dataset is large and diverse, we hypothesized that we could explore families of chemically related materials. For example, lead is a common component of known fluoride-ion conductors, so we examined all Pb-containing systems in our dataset. This yielded 58 different structures, and a small sample of these structures (8 out of 58) is shown in Table 4-1.

Here, we can see Pb-F systems commonly used in FIBs, including PbF_2, PbF_4, and $SnPbF_4$[13,55,56]. We also find many common mixed fluorides, including $BiPb_2F_7$, that have been explored as fluoride conductors[57]. In addition to finding some of the previously known high-performing materials, our search also identified fluoride-ion conductors, such as Pb_3IF_5. This suggests that the strategy of incorporating a small amount of a larger halide may yield many high-performance materials.

Using other query flags, such as anonymous formulas, this approach can be extended to compositionally related systems as well. For example, searching for all structures that have an anonymous formula of AB_3 results in 121 distinct structures. This includes all tysonite LaF3-type crystals (MF_3), where each was predicted to have at least one low-barrier diffusion pathway (<1 eV). This therefore captured all known tysonite fluoride-ion conductors (M=La, Ce, Pr, Nd) and several additional conductors (M=Li, Cu, Y, Dy, Lu, Np, Pu)4. Likewise, searching for all AB_2 structures results in 65 structures. These results include all known fluorite CaF_2-type (MF_2) conductors (M=Cd, Pb, Ba, Ca, Sr) as well as several additional conductors (M=Hg, Eu, Ti), which we list in order of increasing barrier[4]. The trend in barrier height among the known conductors agrees well with experimental measurements (Pb < Ba < Sr < Ca)[58,59,60]. These example searches illustrate the quality of our dataset and how it can assist the exploration of fluoride-ion conductors by composition.

As a second approach for identifying materials, we can explore our dataset by structure-type. For example, in cation shuttle batteries, intercalation electrodes have been used to improve cyclability relative to conversion electrodes. However, in FIBs, conversion-based electrodes are used in the majority of studies, and to our knowledge, only four intercalation structure types have been proposed: Ruddlesden–Popper (K_2NiF_4-type)[61], Schafarzikite (MSb_2O_4)[62], anion-deficient perovskite (y-AMO_3)[63], and layered rocksalts (MoS_2-type)[3]. We therefore searched for layered structures that could facilitate intercalation.

To identify intercalation structures, we searched for layered structure types that have (a) 2-D percolation networks for fluoride and (b) at least three stable structures with a pathway barrier height between 0 and 1 eV. Figure 4-5 categorizes the resulting materials by structure type. None of these six

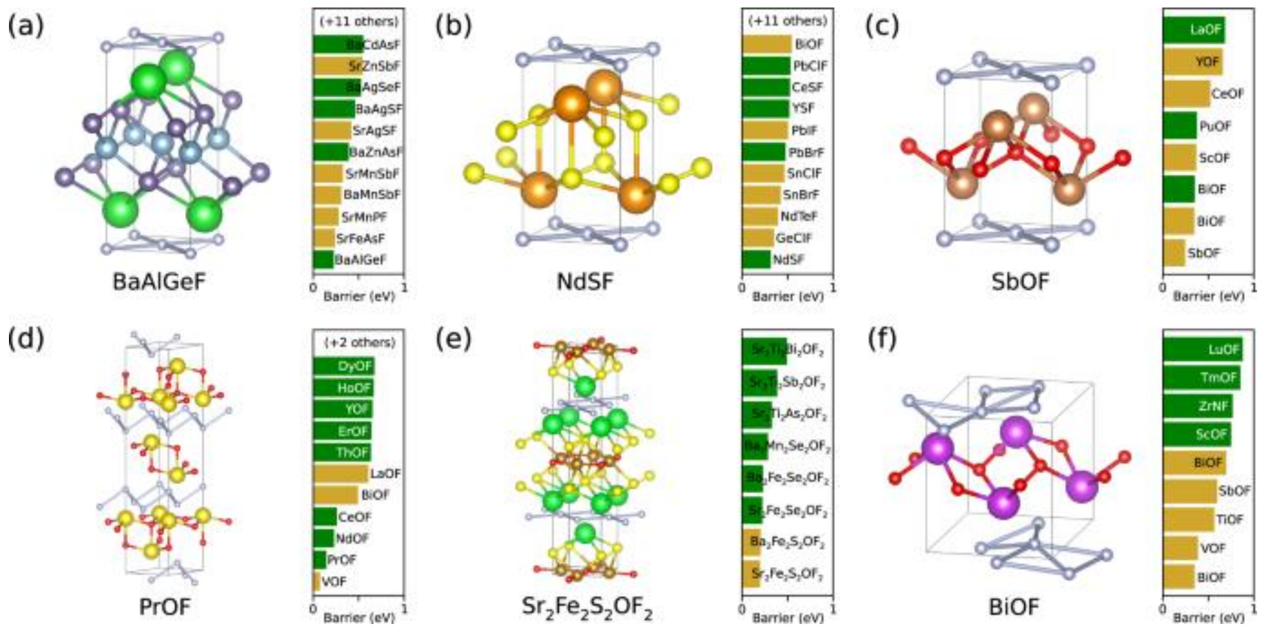


Figure 4-5. Structure types for fluoride intercalation in layered materials. These six structure types contain at least three stable compositions ($E_{hull} = 0$ eV), each with a predicted pathway barrier between 0 and 1 eV. Each structure type shows an example crystal structure and its composition. The corresponding bar charts show barriers of compositions with the same structure. Compositions labeled by a green bar are thermodynamically stable, while gold is metastable according to the Material Project database[40].

structure types have been previously explored for fluoride intercalation. We emphasize that criterion (b) limited the number of structure types that we highlight here; in fact, our dataset includes a very large number of unexplored structures. For example, our search finds the promising material Ho_2CF_2 (MoS_2-type, barrier = 258 meV) but fails to find three or more promising candidates because the Materials Project does not yet include Y_2CF_2 or Sc_2CF_2, which are even more promising as fluoride-intercalation electrodes[3,14,15,16].

A third and final strategy for identifying promising materials would consider the needs of specific applications. Here, we search our dataset for solid-state electrolytes that are stable, inexpensive, and fluoride-conducting. This is accomplished by filtering our dataset with the following conditions: (a) the structure is stable, (b) the bandgap is greater than 3.5 eV, (c) the barrier height is below 700 meV, (d) the raw element cost is less than \$125 per kg and \$125 per mole, (e) the material's percolating network is 3-

D, and (f) the material does not contain mobile cations. This results in 12 structures that are shown in Table 4-2.

These 12 structures include a few well-known fluoride-ion conductors (PbF_2, LaF_3, InF_3, and NdF_3) but primarily consist of materials. The remaining results include promising but unexplored binary and ternary fluorides. The sole binary, GaF_3, possesses a higher barrier and cost but still merits further exploration for fluoride conductivity as either a pure or mixed phase. The results also contain many promising ternary materials that have not been considered fluoride-ion conductors. Among these, we highlight $ZnTiF_6$ and $MgTiF_6$, which are isostructural to InF_3. These two materials have the lowest barriers in this set of results (55 and 324 meV), and the oxidation states of Zn^{2+}, Mg^{2+} and Ti^{4+} suggest that F^{-} will be the only mobile ion. Moreover, the redox stability of Ti^{3+} would facilitate the formation of F^{-} vacancies necessary for conduction.

Table 4-2 Candidate materials for cost-effective solid-state electrolytes.

Material	**Materials project ID**[40]	**E_{approx} (meV) (all paths listed)**	**Has it been synthesized? (COD ID)**[66]	**Is it a known F^{-} conductor?**
NdF_3	mp-18511	−40, 24	Yes (cod-1010985)[a]	Yes[67]
PbF_2	mp-315	542, 1141	Yes (cod-9009027)	Yes[59]
LaF_3	mp-905	564, 637, 684, 750	Yes (cod-9008114)	Yes7
InF_3	mp-6949	618, 652, 1048, 4682	Yes (cod-1535574)	Yes[68]
GaF_3	mp-588	665, 674, 1372, 6218	Yes (cod-8100893)	No
$ZnTiF_6$	mp-1539332	55, 366	No[69] [b]	No
$MgTiF_6$	mvc-14678	324, 435	No[70] [b]	No
$BaZnF_4$	mp-3881	489, 561, 603	Yes (cod-2104457)	No
$BaTiF_6$	mp-8291	500, 533, 978	Yes (cod-1545628)[a]	No
Zr_3InF_{15}	mp-34291	570, 597, 597	No	No
$SrZrF_6$	mp-1208602	585, 836	Yes[71]	Yes[71]

[a] *The COD structure specified is not an exact symmetry match to the Materials Project structure.*
[b] *The material was synthesized as a hexahydrate, but the dehydrated crystal was not reported.*

4.4 Discussion

In this work, we introduced a high-throughput strategy to identify fluoride-ion conductors. An important component of our strategy was the development and validation of a low-cost, high-accuracy DFT method that yielded barrier heights for a large number of fluoride-containing materials. This dataset has allowed us to describe heuristics for the movement of fluoride and to identify many promising structure types for fluoride-ion conductors that have not been previously explored.

The development of FIBs has been limited by the few electrolytes and electrodes known, but our results now give many interesting materials to target for experimental exploration. In the context of electrolytes, some of the most exciting materials are the category $M1M2F_6$ (e.g., ZnTiF6 and MgTiF6), for which there are many low-barrier conductors available from lightweight and inexpensive elements. From the standpoint of electrode development, many of the materials identified here may be suitable for either intercalation or conversion reactions as electrodes. These and many other applications may emerge from this dataset, which is available as an online database for download and searching (see SI).

As an alternative to hierarchical searches, our DDI strategy has demonstrated several advantages. Hierarchical searches must apply cutoff criteria based on initial understandings, but because no preexisting heuristics existed for F^- transport, this approach was not possible. Thus, rather than removing candidates using unvalidated predictors, the DDI strategy allowed us to update the ranking of candidates as our understanding of F^- transport improved. Our DDI strategy therefore enables the exploration of systems where there is little prior knowledge. This approach is certainly not limited to calculations of ionic conduction, and if adopted more widely, it could greatly improve our ability to explore material systems and properties.

4.5 Supporting Information

4.5.1 DDI Priority Queues

Traditionally, high-throughput hierarchical searches are carried out via depth-first or breadth-first searches (Figure 4-S1, top-left/top-right), whereas we orchestrated calculations based on dynamic, decoupled, and iterative priority queues (Figure 4-S1, bottom). Depth-first searches carry out all calculations for a single structure within a single workflow, where it is analyzed to completion or until it fails an intermediate check (e.g., DFT is skipped if the empirical method predicts a high barrier). Thus, all candidate materials are submitted to a fixed queue at the start of a study. Breadth-first searches run a specific analysis on all candidate structures, after which a cutoff is selected to determine which candidates continue (e.g., once all empirical analysis is completed, only the top 10% of candidates continue to the next level). Thus, candidates are submitted at each stage, where priority can be adjusted between stages but not after submission. The strategy used in this study removes any cutoff criteria and prerequisites for each structure and calculation level, allowing calculations to be completed via a flexible priority queue. Rather than submitting candidates to a queue, candidates are requested by task executors in order of priority. Reranking of candidates occurs while task executors are requesting new candidates for a calculation. We updated the ranking of candidates throughout our study, where only illustrative examples were discussed in the main text (e.g., our priority based on incorrect ΔIRO was replaced with a reprioritized using our updated ΔIRO model).

4.5.2 DFT Parameters

Ab initio calculations used with VASP[72] with the Materials Project's precursor input settings (MIT relaxation and NEB sets)[32]. Error handling used Custodian[41] with standard, lightweight error handlers associated with MD calculations in Atomate[32]. For static calculations, we reduced the electronic convergence criteria to $1x10^{-3}$ eV for a moderate increase in throughput.

4.5.3 Supercell Pathway Initialization

Prior to any analysis, structures were reduced to primitive cells at 0.1 Å tolerance and then converted to an LLL lattice-basis[73]. The supercell structures were generated by scaling sanitized structures until each lattice vector was a minimum length (R), e.g., R=10 Å for NEB and R=7 Å for all other analyses described in the main text. For all calculations, pathways were prerelaxed by IDPP.[42] This was performed on all sites in the supercell to avoid the common assumption of a rigid host lattice.

4.5.4 Dimensional Analysis

Dimensionality of both host lattices and percolating pathways were determined via the Larson scoring parameter with the most probable value being reported.[74] Atomic connectivity of host lattices was determined by removing all fluoride in the structure and then analyzing with the CystralNN algorithm.[75] Pathway connectivity was determined by two methods for each individual path: (1) dimensionality of the individual diffusion pathway when including all symmetrically-equivalent paths and (2) dimensionality when including all pathways of equal or shorter length. On a single-pathway basis, only 17% (7,226) were determined to be percolating, whereas this increased to 42.8% (18,569) if paths of equal or shorter length were included. For each crystal, the Larson scoring parameter was also used to determine the dimensionality of overall fluoride diffusion networks (as described in the following section).

4.5.5 Barrier Height for Long-range Transport

The preceding section showed that long-range transport (i.e., percolative transport) can vary in its complexity. In the simplest case, long-range transport can occur when a fluoride achieves percolative transport with a single type of hop. For this case, the barrier for long-range transport is the same as the barrier for the single hop (Figure 4-S2, top). In more complex cases, percolation requires two or more different types of hops–that is, long-range transport requires the use of two or more symmetrically distinct pathways. For this case, the barrier for long-range transport is equal to the energy difference between the highest and lowest energies along the overall route (Figure 4-S2, bottom). In a material that has multiple percolative routes, we report the lowest barrier for long-range transport. Results are provided in "percolating_networks.csv". Fluoride mobile structures (overall $E_{approx} < 1$ eV) that are described by one symmetrically unique pathway occur more frequently in our dataset (1,203 structures) compared to fluoride mobile structures that require more than one unique pathway to be percolating (56 structures). This is consistent with our input structures being primarily high-symmetry, single-crystal structures.

4.5.6 Structure Type Matching

We started by matching each structure to the AFLOW Encyclopedia of Crystallographic Prototypes[76,77,78]; however, this only matches 20% structures. We therefore performed anonymous matching with all remaining structures at a tolerance of 0.1 Å, yielding 1,207 unique structure-type groupings that contained at least two equivalent structures.

4.5.7 Cost Analysis

The prices for compositions were predicted using pymatgen's cost database, which uses raw element pricing that is aggregated from several source webpages.[79,80] Using their sources, prices were updated, and we used prices from the "Prices of Chemical Elements" table when there was disagreement between costs.[80] The original sources of the aggregated data are provided on the corresponding webpages.

4.5.8 Supporting Figures and Table

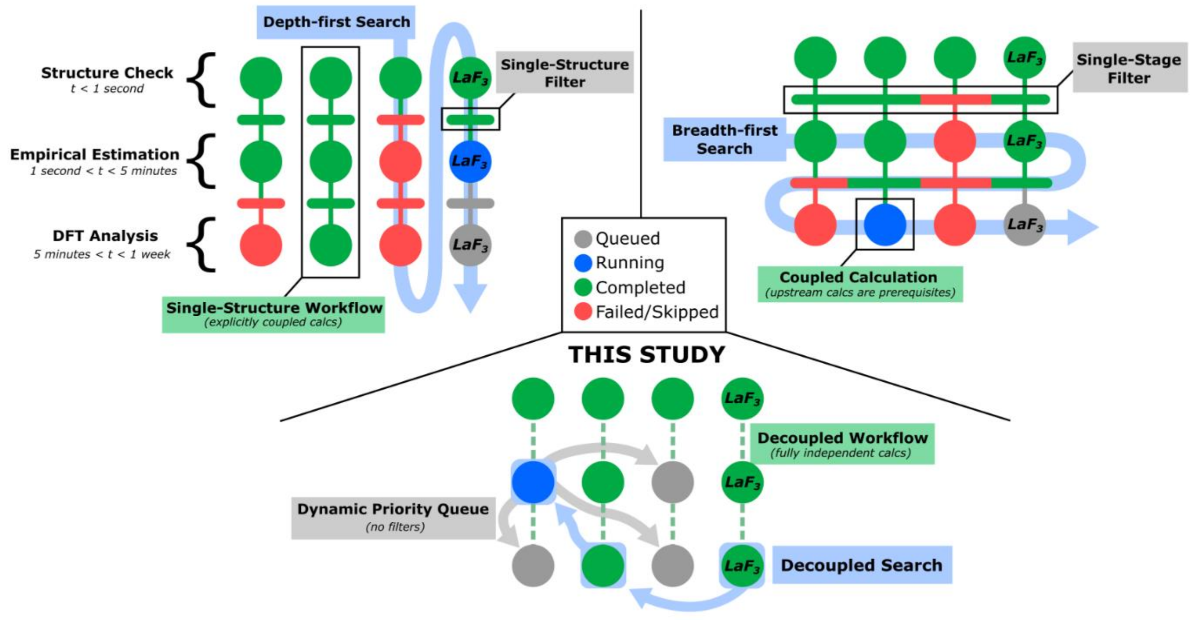


Figure 4-S1. Illustration of our dynamic priority queue (bottom) in comparison to commonly applied depth-first (top-left) and breadth-first (top-right) searches. Hierarchical calculations are composed of different calculation “levels” of increasing quality and computational expense, where three levels are shown in this schematic (Structure Check, Empirical Estimation, and DFT Analysis).

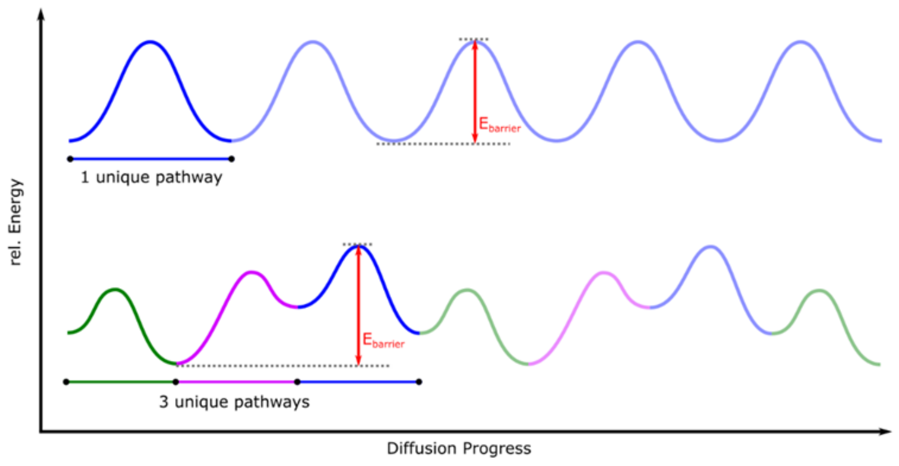


Figure 4-S2. Example energy profiles for long-range diffusion. (top) Percolation occurs via a network of pathways hops that are symmetrically equivalent. (bottom) Percolation occurs via a network of pathways that include multiple unique hops.

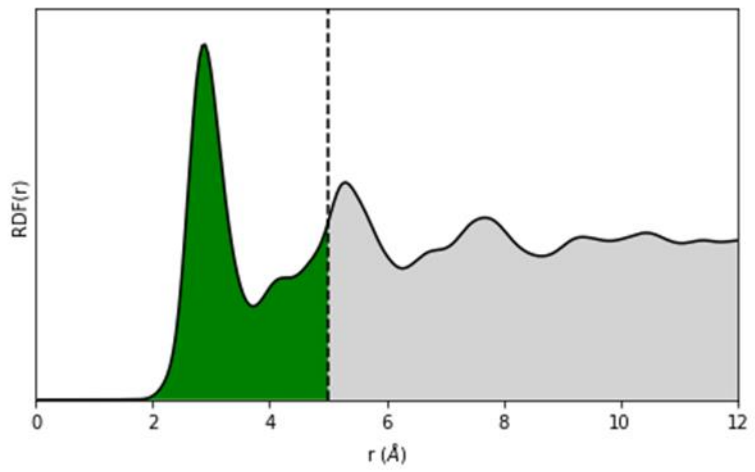


Figure 4-S3. The cumulative fluorine-fluorine (F-F) radial distribution function (RDF) for all structures in this study. Prior to summing RDFs across all structures, individual-structure RDFs are normalized to $RDF(r)_{max}=1$ as well as gaussian filtered to absorb finite differences across structures. Only F-F pairings of up to 5 Å (green) are considered for vacancy diffusion in this study, while those greater than 5 Å (grey) are excluded.

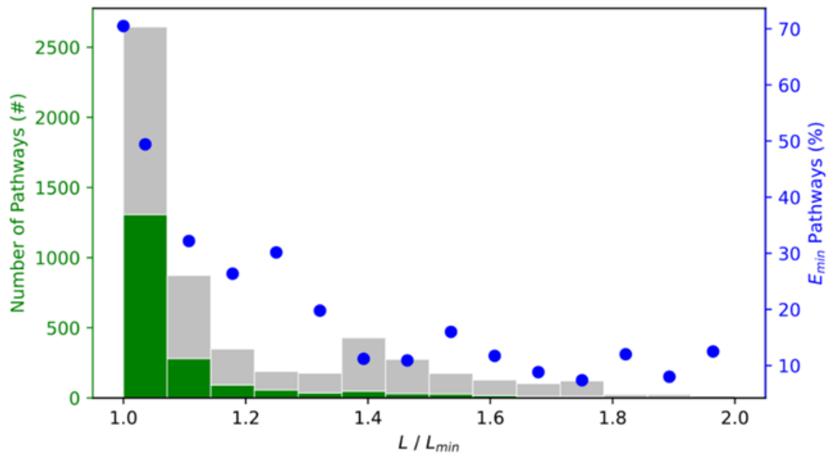


Figure 4-S4. Pathway length (L) relative to the shortest fluoride pathway (L_{min}) in a structure. The lowest barrier pathway in each structure (the pathway in which $E=E_{min}$) is indicated in green and all other pathways are in grey. In 70.5% of structures, the shortest pathway of the structure also has the lowest barrier. Within a structure, as the difference in path lengths becomes larger, the percent likelihood that the

longer pathway has a lower barrier decreases exponentially (blue). Only those structures with at least two DFT-calculated barriers are included.

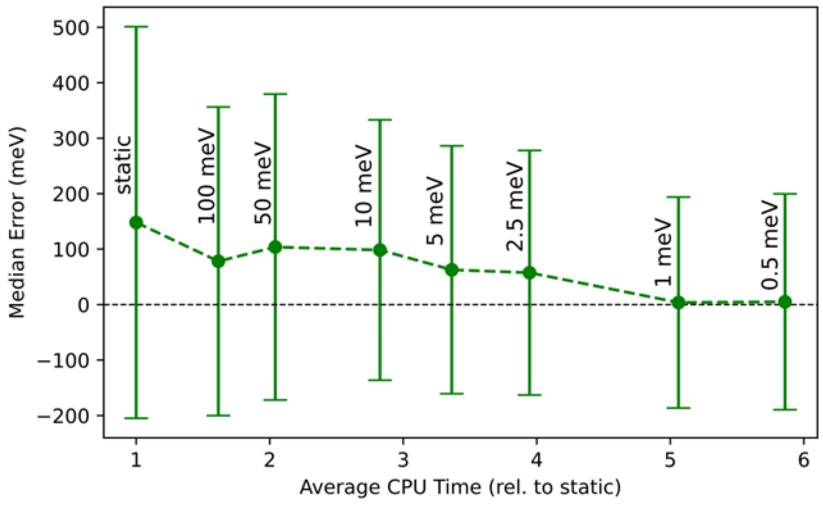


Figure 4-S5. Relaxation of pathways under variable convergence criteria. Relaxations were run to an ionic convergence of 0.5 meV, and individual ionic steps were analyzed for accuracy and CPU usage at different stages of convergence. This small supercell calculation (R=7 Å) was compared to reference NEB barriers (R=10 Å), where error bars are the standard deviation in barrier error. The first datapoint (static) is exactly equivalent to our initial static approximation, while full relaxation (0.5 meV) is approximately equivalent to our reference dataset (R=10 Å). This allows us to estimate that the use of a 7 Å supercell rather than a 10 Å supercell increases barrier heights by approximately 200 meV. We remind the reader that our linear regression method corrects for this systematic error.

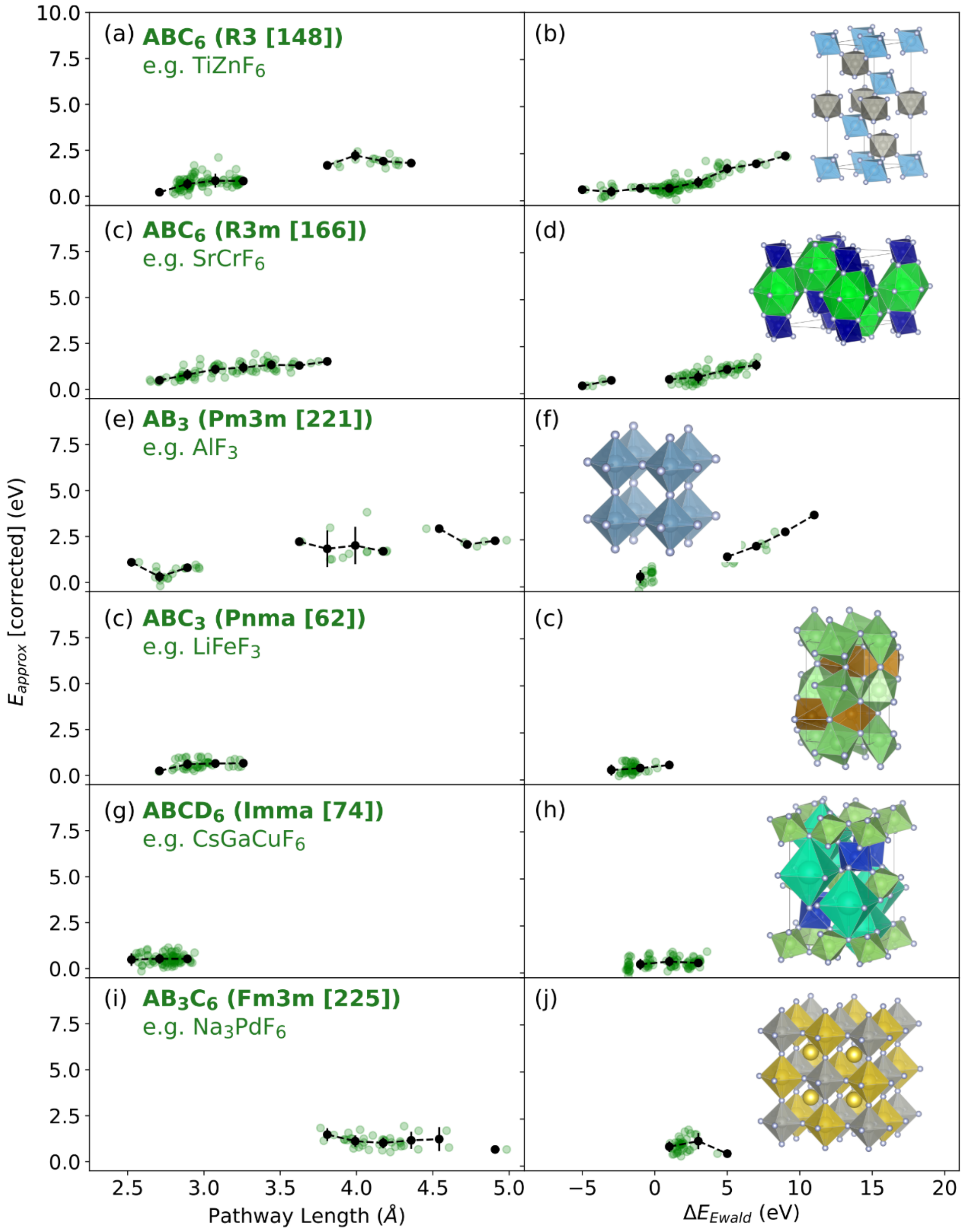


Figure 4-S6. (Continues on next page)

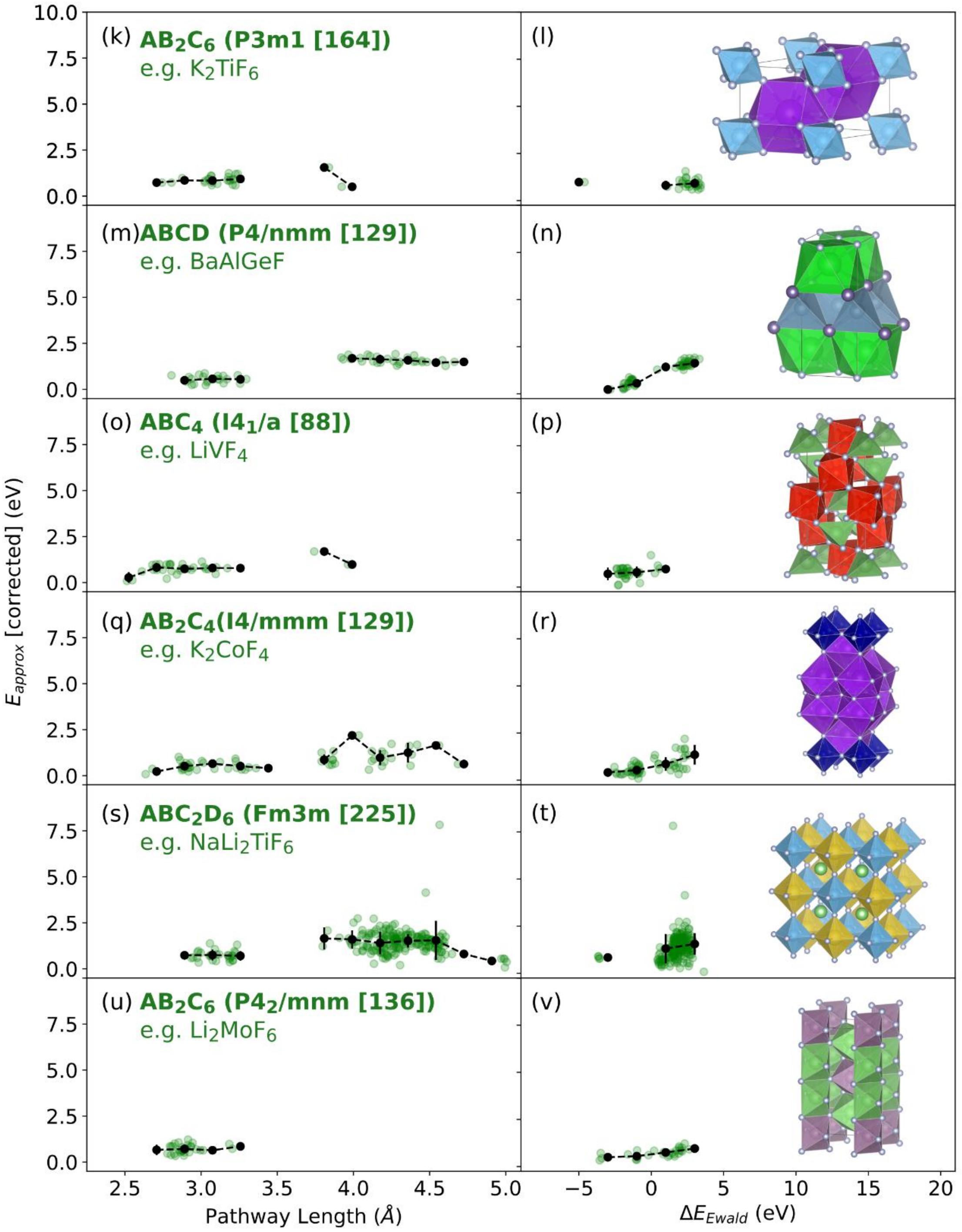


Figure 4-S6. Simple predictors for fluoride mobility based on pathway length and change in Ewald energy (ΔE_{Ewald}). A subset of common structure types are presented, where each row is a single structure type – labeled with an anonymous formula, example composition, and example crystal structure from the structure type. This figure is an extension of Figure 4-4 in the main text.

Table 4-S1. All converged midpoint-only NEB calculations. Note that these barriers represent the minimum possible barriers for each pathway. Negative E_{NEB} occurs in pathways where start/end structures are not representative of the true ground state, even after relaxation. This data (along with additional table headers) is also available in the CSV format in pathway_calculations.csv.

Pathway ID	*Structure ID*	*Reduced Formula*	*E_{hull} (ev)*	*Pathway Dimensionality*	*E_{approx} (corrected) (eV)*	*E_{NEB} (eV)*
594	mp-10175	$KCdF_3$	0.026	3	0.415	-1.090
869	mp-989178	ScOF	0.084	2	0.366	-0.628
293	mp-1216625	$TiOF_2$	0.048	2	0.306	-0.537
151	mp-973778	Hg_3F	0.162	3	0.225	-0.383
6625	mp-556194	$SrSbSe_2F$	0.011	2	0.108	-0.353
633	mp-998358	$MgAgF_3$	0.033	3	0.301	-0.178
1230	mp-1216550	$TlBiF_4$	0.010	2	0.381	-0.084
3	mp-2175	TlF	0.036	3	0.605	-0.028
18924	mp-27355	Pb_2OF_2	0.000	1	0.450	-0.024
1184	mvc-11360	BiOF	0.001	1	0.340	-0.013
8764	mp-1103397	$BaSnF_4$	0.000	2	0.292	-0.013
1565	mp-8962	Rb_2HgF_4	0.000	2	0.329	-0.011
8701	mp-1210620	LuSeF	0.019	2	-0.053	-0.001
1046	mp-5606	$AlTlF_4$	0.002	2	0.815	0.004
10373	mp-1207668	TmSeF	0.000	2	-0.057	0.008
9924	mp-3632	YSF	0.026	2	0.114	0.008
1563	mp-555850	LuSF	0.000	2	0.002	0.023
11004	mp-1212164	$Hg_3(SeF)_2$	0.000	3	0.462	0.031
1198	mp-753594	BiOF	0.000	2	0.350	0.033
1113	mp-556646	$RbFeF_4$	0.016	2	0.230	0.037
2647	mp-758604	$LiAgF_2$	0.054	3	0.246	0.064
8685	mp-27167	Sn_2IF_3	0.000	1	0.340	0.077
71	mp-8177	HgF_2	0.000	3	0.180	0.086
8383	mp-27373	SnClF	0.000	1	0.402	0.101
24817	mp-16915	$BaTiOF_4$	0.000	1	0.358	0.101
30390	mp-555048	$NaAlCdF_6$	0.000	1	0.307	0.102
109	mp-706	HgF	0.000	2	0.348	0.103
3857	mp-1210270	Na_2YF_6	0.000	2	0.331	0.105
659	mp-7482	$RbHgF_3$	0.000	3	0.265	0.117
674	mp-9006	Ho_2CF_2	0.000	2	0.258	0.133
1044	mp-5606	$AlTlF_4$	0.002	2	0.555	0.137
389	mp-558852	FeF_3	0.011	3	0.238	0.139
14150	mp-1209528	$PdPb_2F_6$	0.000	1	0.160	0.141
7852	mp-1021492	Cd_4SF_6	0.028	2	0.171	0.156
754	mp-13819	$KAgF_3$	0.000	3	0.420	0.161
14151	mp-1209528	$PdPb_2F_6$	0.000	1	0.157	0.165
368	mp-998761	$TlNiF_3$	0.011	3	0.440	0.167
32573	mp-998758	$TlHgF_3$	0.000	1	0.303	0.171
1258	mp-10931	HoSF	0.000	2	0.547	0.178
1515	mp-1025338	Tl_2CuF_4	0.000	2	0.298	0.179
1052	mp-10086	YSF	0.000	2	0.521	0.184
3856	mp-1210270	Na_2YF_6	0.000	1	0.322	0.185
80	mp-241	CdF_2	0.000	3	0.239	0.198
2719	mp-1080135	SrZnAsF	0.000	2	0.569	0.198
923	mp-974603	HgF_2	0.008	3	0.328	0.200
27922	mp-18832	$K_2VO_2F_3$	0.000	1	0.152	0.200
22328	mp-905	LaF_3	0.000	0	0.564	0.205
11844	mp-7580	$Hg_3(SF)_2$	0.000	3	0.516	0.208
888	mvc-16450	$CaSnF_4$	0.043	2	0.518	0.215
7860	mp-1021492	$Cd4SF_6$	0.028	2	0.229	0.215
20184	mp-1013726	Cd_4SF_6	0.008	1	0.265	0.218
19202	mp-9628	$KCdF_3$	0.000	1	0.466	0.227
2055	mp-555014	Rb_2MnF_5	0.000	1	0.208	0.230
2228	mp-6949	InF_3	0.000	2	0.618	0.238
5486	mp-4360	$NaCdF_3$	0.000	1	0.501	0.239
1537	mp-1147757	$CuPb_2(OF)_2$	0.117	2	0.182	0.239

10891	mp-19956	Ba_2InO_3F	0.000	2	0.538	0.241
8967	mp-755244	$LiVF_4$	0.103	3	0.400	0.243
20704	mp-28855	Cd_4OF_6	0.000	2	0.281	0.246
32575	mp-998758	$TlHgF_3$	0.000	1	0.293	0.254
8021	mp-7386	$NaAgF_4$	0.000	2	0.473	0.254
4	mp-7592	AgF	0.000	3	0.403	0.262
2981	mp-1880	SbF_3	0.000	1	0.572	0.265
8328	mp-753257	$LiCuF_4$	0.000	1	0.594	0.266
19201	mp-9628	$KCdF_3$	0.000	1	0.504	0.267
1354	mp-10930	$TbSF$	0.000	2	0.547	0.267
22193	mp-867665	$LiV_3(OF_3)_2$	0.000	1	0.188	0.267
7889	mp-998422	$CuAgF_3$	0.013	3	0.325	0.279
20281	mp-17972	$Zn_2Hg_2OF_6$	0.000	3	0.464	0.284
9196	mp-765559	$LiAgF_4$	0.000	1	0.473	0.285
20334	mp-17745	$Mn_2Hg_2SF_6$	0.465	3	0.422	0.288
20707	mp-28855	Cd_4OF_6	0.000	1	0.318	0.290
22327	mp-905	LaF_3	0.000	0	0.637	0.291
14004	mp-1105287	$Ti_4Pb_2O_9F_2$	0.007	1	0.310	0.296
1120	mp-3931	$SmSF$	0.000	2	0.551	0.298
22805	mp-555667	$RbFeF_4$	0.000	1	0.219	0.306
19805	mp-31132	Hg_4OF_6	0.000	1	0.357	0.307
9862	mp-776264	$LiFeF_4$	0.000	1	0.205	0.310
9504	mp-27175	$InOF$	0.000	1	0.156	0.313
4649	mp-1078631	$SrBiS_2F$	0.009	2	0.124	0.313
560	mp-6951	$RbCdF_3$	0.000	3	0.454	0.315
9488	mp-2284	AgF_2	0.009	1	0.338	0.320
39	mp-1391	Ag_2F	0.000	2	0.400	0.320
10892	mp-19956	Ba_2InO_3F	0.000	2	0.941	0.321
2631	mp-1080514	SrF_3	0.000	1	0.252	0.329
4668	mp-998418	$MgAgF_3$	0.018	2	0.353	0.332
32946	mp-776692	$LiFe_2F_7$	0.007	1	0.234	0.332
2001	mp-752467	NbO_2F	0.000	1	0.168	0.333
9869	mp-1176702	$LiFeF_4$	0.005	1	0.221	0.339
3178	mp-22398	FeF_3	0.007	3	0.205	0.347
1750	mp-1206532	Tl_2NiF_4	0.000	2	0.343	0.348
2229	mp-6949	InF_3	0.000	3	0.652	0.355
15221	mp-556497	$RbNi_2F_6$	0.042	3	0.305	0.357
1322	mp-23008	$PbBrF$	0.000	2	0.479	0.357
4666	mp-998418	$MgAgF_3$	0.018	3	0.371	0.367
18221	mp-3795	$NaZnF_3$	0.000	1	0.571	0.367
3998	mp-551403	$Ba_2Fe_2S_2OF_2$	0.005	2	0.204	0.368
8226	mp-1206495	Pb_2BrOF	0.000	1	0.411	0.369
16537	mp-757118	$LiMn_2F_6$	0.024	2	0.347	0.371
32103	mp-34291	Zr_3InF_{15}	0.000	0	0.571	0.378
9842	mp-560449	$K_3Ni_2F_7$	0.000	2	0.561	0.379
2941	mp-549058	$Ba_2Fe_2Se_2OF_2$	0.000	2	0.229	0.379
4179	mp-1206410	Pb_2IO_F	0.000	1	0.388	0.382
1517	mp-1025338	Tl_2CuF_4	0.000	2	0.312	0.383
485	mp-5566	$KCuF_3$	0.000	3	0.389	0.384
13051	mp-756285	V_2OF_5	0.001	1	0.156	0.385
6301	mp-7767	$RbAgF_3$	0.000	1	0.458	0.386
1910	mp-1079320	$BaZnAsF$	0.000	2	0.392	0.387
23246	mp-777875	$LiFeF_4$	0.009	1	0.255	0.390
12802	mp-559450	FeF_3	0.011	3	0.437	0.392
8524	mp-554517	$NaMnF_4$	0.000	1	0.356	0.396
786	mp-5878	$KZnF_3$	0.000	3	0.527	0.397
3470	mp-541384	MnF_3	0.026	2	0.339	0.399
2338	mp-1095151	$BaCdAsF$	0.000	2	0.551	0.402
22932	mp-559739	$RbVOF_3$	0.000	1	0.375	0.404
2464	mp-556560	MnF_3	0.000	1	0.356	0.408
3489	mp-541384	MnF_3	0.026	3	0.344	0.410
7679	mp-3682	$KCuF_3$	0.000	2	0.361	0.413
2688	mp-1086652	$BaAgSF$	0.000	2	0.461	0.414
2329	mp-1078562	$SrAgSF$	0.005	2	0.420	0.416
2075	mp-1079647	$BaAgSeF$	0.000	2	0.528	0.418
2772	mp-752927	$LiAgF_2$	0.040	2	0.225	0.418
3068	mp-1080029	$Ba_2Mn_2Se_2OF_2$	0.000	2	0.283	0.418
3933	mp-1079717	$Sr_2Ti_2Sb_2OF_2$	0.000	2	0.385	0.423
3599	mp-1079747	$Sr_2Ti_2As_2OF_2$	0.000	2	0.327	0.424
1063	mp-759883	$BiOF$	0.009	2	0.493	0.424

9361	mp-8161	$CaPdF_4$	0.000	2	0.510	0.425
536	mp-555036	$KCrF_3$	0.087	3	0.583	0.441
9838	mp-560449	$K_3Ni_2F_7$	0.000	2	0.448	0.441
7178	mp-23104	Pb_2ClOF	0.000	1	0.437	0.447
26	mp-1009009	LiF	0.288	3	0.392	0.447
1866	mp-752931	Li_2MnF_4	0.086	2	0.198	0.449
30388	mp-555048	$NaAlCdF_6$	0.000	1	0.505	0.450
11007	mp-757235	Li_2CuF_4	0.031	3	0.420	0.458
2462	mp-556560	MnF_3	0.000	1	0.355	0.462
7363	mp-8858	$RbCuF_3$	0.000	2	0.527	0.464
22400	mp-27210	$RbTlF_4$	0.000	1	0.412	0.467
1154	mp-7715	AgF_2	0.000	1	0.557	0.471
660	mp-7482	$RbHgF_3$	0.000	2	0.888	0.472
19964	mp-504704	$Ni_2Hg_2OF_6$	0.000	3	0.632	0.476
3232	mp-1078216	$Sr_2Ti_2Bi_2OF_2$	0.000	2	0.492	0.477
31516	mp-558123	$V_2Ge(O_2F)_2$	0.000	1	0.342	0.478
1948	mp-1078949	BaMnSbF	0.009	2	0.309	0.479
603	mp-1099572	$MnInF_3$	0.031	3	0.543	0.480
8484	mp-776164	$LiFeF_4$	0.005	1	0.322	0.480
10312	mp-558059	$LiMnF_4$	0.000	1	0.438	0.485
1686	mp-1078453	BaAlGeF	0.000	2	0.228	0.488
3437	mp-1080128	SrMnSbF	0.044	2	0.328	0.491
13776	mp-1208602	$SrZrF_6$	0.000	0	0.836	0.492
3908	mp-1080828	$KCuF_3$	0.001	1	0.296	0.492
23256	mp-777875	$LiFeF_4$	0.009	1	0.335	0.494
931	mp-5394	LaSF	0.000	2	0.594	0.496
3339	mp-588	GaF_3	0.000	2	0.665	0.497
1981	mp-562468	TiF_3	0.002	3	0.485	0.504
3199	mp-1078693	BaCdSbF	0.000	2	0.557	0.505
23765	mp-8892	$LiInF_4$	0.000	1	0.577	0.507
122	mp-246	TiF_3	0.000	3	0.479	0.510
528	mp-341	PbF_4	0.000	2	0.771	0.513
39097	mp-558599	$K_2Mn_2P_2O_7F_2$	0.000	1	1.677	0.515
31060	mp-1194849	Na_3MoO_4F	0.000	1	0.376	0.518
657	mp-554973	$TlCuF_3$	0.000	3	0.348	0.518
22326	mp-905	LaF_3	0.000	0	0.684	0.523
22291	mp-28778	$KScF_4$	0.000	1	0.613	0.523
9633	mp-3125	$K_3Zn_2F_7$	0.000	2	0.562	0.525
9409	mp-759601	MnOF	0.029	3	0.423	0.531
2091	mp-1080650	BaCuSF	0.004	2	0.668	0.534
2904	mp-559931	VF_3	0.000	2	0.321	0.541
9630	mp-3125	$K_3Zn_2F_7$	0.000	2	0.570	0.545
3338	mp-588	GaF_3	0.000	3	0.674	0.547
32927	mp-998759	$TlCdF_3$	0.000	1	0.573	0.548
10319	mp-558059	$LiMnF_4$	0.000	1	0.453	0.554
1569	mp-8962	Rb_2HgF_4	0.000	2	0.322	0.563
22938	mp-559739	$RbVOF_3$	0.000	1	0.612	0.568
13608	mp-780857	Li_2VOF_4	0.000	1	0.465	0.569
11471	mp-753202	Li_5CuF_8	0.002	2	0.335	0.578
2837	mp-29764	$RbHF_2$	0.000	1	1.566	0.585
1873	mp-557610	K_2MnF_4	0.000	2	0.698	0.593
22329	mp-905	LaF_3	0.000	0	0.749	0.598
1980	mp-37473	$TiOF_2$	0.079	3	0.416	0.601
2916	mp-559931	VF_3	0.000	3	0.361	0.605
3186	mp-12444	SrCuSF	0.000	2	0.759	0.607
17926	mp-1105117	$Ni_3Sb_4(OF)_6$	0.003	3	0.455	0.615
12054	mp-2632	TlF_3	0.000	1	0.497	0.615
18855	mp-505066	$NaFeF_3$	0.000	1	0.460	0.623
72	mp-8177	HgF_2	0.000	3	1.034	0.633
3231	mp-13287	BaCuTeF	0.000	2	0.734	0.633
527	mp-341	PbF_4	0.000	2	0.497	0.643
1152	mp-1227733	$BaSnF_4$	0.058	2	0.468	0.647
10431	mp-3881	$BaZnF_4$	0.000	1	0.561	0.649
3504	mp-9195	BaCuSeF	0.000	2	0.686	0.663
33893	mp-1214575	$Ba_5Mn_3O_{12}F$	0.000	1	0.425	0.671
8963	mp-755244	$LiVF_4$	0.103	3	0.410	0.692
1231	mp-1216550	$TlBiF_4$	0.010	2	1.113	0.694
7149	mp-1111671	K_2LiInF_6	0.000	2	1.647	0.697
10434	mp-3881	$BaZnF_4$	0.000	1	0.489	0.698
1103	mvc-14177	$MgTiF_4$	0.207	2	0.617	0.717

2764	mp-557539	MoF_3	0.000	2	1.000	0.730
2649	mp-1080438	SrAgTeF	0.000	2	0.731	0.741
1097	mp-22964	PbClF	0.000	2	0.533	0.748
27180	mp-1208058	$TlBi_3F_{10}$	0.000	3	0.769	0.764
1456	mp-8226	ThNF	0.000	2	0.634	0.783
13779	mp-1208602	$SrZrF_6$	0.000	0	0.585	0.797
1286	mp-3637	YOF	0.000	2	0.652	0.801
1096	mp-1072193	ErOF	0.000	2	0.640	0.810
7312	mp-1111108	K_2NaAsF_6	0.000	0	0.469	0.819
534	mp-560976	$KNiF_3$	0.000	3	0.393	0.820
34685	mp-1228113	$Ba_5P_3O_{12}F$	0.000	1	0.417	0.825
1199	mp-753594	BiOF	0.000	2	1.263	0.826
971	mp-1072114	HoOF	0.000	2	0.666	0.832
77	mp-315	PbF_2	0.000	3	0.542	0.836
3513	mp-4829	Na_2ThF_6	0.000	2	0.934	0.844
1173	mp-1072208	DyOF	0.000	2	0.674	0.847
78	mp-315	PbF_2	0.000	3	1.141	0.850
1008	mp-14093	TbOF	0.000	2	0.684	0.865
1540	mp-8111	LaOF	0.005	2	0.608	0.865
20709	mp-28855	Cd_4OF_6	0.000	2	0.882	0.879
1316	mp-5479	$RbAlF_4$	0.001	2	0.790	0.880
10221	mp-754589	VOF	0.000	3	0.712	0.884
2643	mp-16742	BaAgTeF	0.000	2	0.764	0.901
17	mp-504731	EuF_2	0.000	3	0.696	0.912
9587	mp-754758	Li_2FeO_2F	0.003	2	0.162	0.914
1233	mp-1220682	$NaYF_4$	0.001	2	0.684	0.915
1454	mp-1221036	$NaErF_4$	0.002	2	0.708	0.924
1366	mp-1220708	$NaHoF_4$	0.001	2	0.685	0.928
2905	mp-560338	CrF_3	0.000	3	0.720	0.932
1107	mp-7100	LaOF	0.000	2	0.684	0.938
234	mp-10694	ScF_3	0.000	3	0.780	0.952
892	mvc-16450	$CaSnF_4$	0.043	2	1.093	0.954
5398	mp-1111927	K_2LiCrF_6	0.000	0	0.603	0.959
9959	mp-1209982	$NaMoO_3F$	0.000	1	0.557	0.965
1870	mp-557610	K_2MnF_4	0.000	2	0.797	1.007
9588	mp-754758	Li_2FeO_2F	0.003	2	0.347	1.009
76	mp-8177	HgF_2	0.000	3	1.124	1.033
11845	mp-7580	$Hg_3(SF)_2$	0.000	3	1.047	1.038
340	mp-3654	$RbCaF_3$	0.000	3	0.854	1.047
9634	mp-3125	$K_3Zn_2F_7$	0.000	2	0.900	1.062
4845	mp-1113712	Rb_2AgPdF_6	0.000	2	1.342	1.068
28	mp-1029	BaF_2	0.000	3	0.788	1.095
747	mp-2706	SnF_4	0.000	2	0.884	1.096
837	mp-998739	$MgTlF_3$	0.000	3	0.805	1.101
41	mp-2741	CaF_2	0.000	3	0.813	1.104
26410	mp-2943	KY_3F_{10}	0.000	3	1.005	1.114
40	mp-981	SrF_2	0.000	3	0.817	1.138
26538	mp-1211612	KDy_3F_{10}	0.000	3	1.036	1.141
18463	mp-27714	H_3OF	0.000	1	1.306	1.141
1725	mp-9583	K_2ZnF_4	0.000	2	0.869	1.143
1614	mp-31212	K_2MgF_4	0.000	2	0.847	1.157
3398	mp-13339	EuCuTeF	0.009	2	0.795	1.158
24785	mp-1211478	KHo_3F_{10}	0.000	3	1.054	1.164
2	mp-11718	RbF	0.000	3	1.028	1.176
977	mp-1873	ZnF_2	0.000	1	1.234	1.188
33762	mp-6669	$Sr_5P_3O_{12}F$	0.000	1	0.611	1.191
9852	mp-560449	$K_3Ni_2F_7$	0.000	2	0.592	1.195
1875	mp-557610	K_2MnF_4	0.000	2	0.953	1.226
2585	mp-1091416	SrCuTeF	0.018	2	0.871	1.239
9754	mp-554144	$K_3Mn_2F_7$	0.000	2	0.900	1.256
9110	mp-34081	$NaYF_4$	0.000	1	1.685	1.260
3772	mp-1111110	K_2LiScF_6	0.000	2	1.904	1.260
11	mp-463	KF	0.000	3	1.077	1.298
4032	mp-560963	Rb_2NaMoF_6	0.000	2	1.750	1.309
1403	mp-555934	VF_2	0.000	3	1.113	1.331
20490	mp-15073	$BaSmC_2O_6F$	0.000	2	0.973	1.379
832	mp-5347	$KAlF_4$	0.017	2	1.020	1.400
1319	mp-5479	$RbAlF_4$	0.001	2	1.007	1.408
1056	mp-22951	BaIF	0.000	2	0.899	1.442
1335	mp-556382	EuClF	0.000	2	0.921	1.463

13695	mp-29946	IO_2F	0.000	1	2.190	1.485
925	mp-23046	SrIF	0.000	2	0.965	1.490
8	mp-682	NaF	0.000	3	1.081	1.501
7458	mp-1111112	K_2LiGaF_6	0.000	2	2.107	1.502
897	mp-3448	$KMgF_3$	0.000	3	1.022	1.508
31422	mp-6280	$Al_2Si(O_2F)_2$	0.000	1	1.235	1.548
3341	mp-588	GaF_3	0.000	1	1.372	1.587
7263	mp-1113857	Rb_2AlAgF_6	0.000	2	2.002	1.595
21970	mp-10234	Sr_2BN_2F	0.000	1	1.338	1.607
26320	mp-6160	Rb_2PO_3F	0.000	1	1.267	1.662
349	mp-3654	$RbCaF_3$	0.000	2	1.444	1.692
1949	mp-7604	Mg_3NF_3	0.000	3	1.219	1.698
1397	mvc-3135	$TiZnF_4$	0.346	2	0.467	1.721
214	mp-24199	$LiHF_2$	0.000	2	1.458	1.782
5400	mp-1111927	K_2LiCrF_6	0.000	2	1.903	1.895
3233	mp-1078216	$Sr_2Ti_2Bi_2OF_2$	0.000	2	1.626	2.002
53	mp-2741	CaF_2	0.000	3	1.790	2.018
32278	mp-1195159	Tb_2SeOF_2	0.000	1	1.814	2.024
1805	mp-557938	$HfVF_6$	0.000	2	2.034	2.030
1296	mp-3637	YOF	0.000	2	2.216	2.045
513	mp-557257	KVF_3	0.000	2	1.633	2.046
213	mp-24199	$LiHF_2$	0.000	2	1.272	2.055
123	mp-246	TiF_3	0.000	2	1.560	2.061
1174	mp-1072208	DyOF	0.000	2	2.240	2.085
5454	mp-560936	Rb_2NaCrF_6	0.000	2	1.818	2.161
5490	mp-1114702	Rb_2LiScF_6	0.000	2	2.176	2.303
13103	mp-11166	$BaZnCO_3F_2$	0.000	2	2.634	2.319
118	mp-282	TiF_2	0.248	3	1.077	2.323
1034	mp-9488	SmOF	0.000	2	1.877	2.330
930	mp-23046	SrIF	0.000	2	2.014	2.454
1485	mp-23070	BaBrF	0.000	2	2.318	2.493
1183	mp-7738	LaSeF	0.000	2	2.352	2.496
2839	mp-29764	$RbHF_2$	0.000	1	2.808	3.026

REFERENCES

(1) Biemolt, J., Jungbacker, P., van Teijlingen, T., Yan, N. & Rothenberg, G. Beyond lithium-based batteries. *Materials* **13**, 425 (2020).

(2) Karkera, G., Reddy, M. A. & Fichtner, M. Recent developments and future perspectives of anionic batteries. *J. Power Sources* **481**, 228877 (2021).

(3) Druffel, D. L. et al. First-principles prediction of electrochemical electron-anion exchange: ion insertion without redox. *J. Phys. Chem. Lett.* **11**, 9210–9214 (2020).

(4) Nowroozi, M. A. et al. Fluoride ion batteries – past, present, and future. *J. Mater. Chem. A* **9**, 5980–6012 (2021).

(5) Zhao, X., Zhao-Karger, Z., Fichtner, M. & Shen, X. Halide-based materials and chemistry for rechargeable batteries. *Angew. Chem. Int. Ed.* **59**, 5902–5949 (2020).

(6) Gschwind, F. et al. Fluoride ion batteries: theoretical performance, safety, toxicity, and a combinatorial screening of new electrodes. *J. Fluor. Chem.* **182**, 76–90 (2016).

(7) Anji Reddy, M. & Fichtner, M. Batteries based on fluoride shuttle. *J. Mater. Chem.* **21**, 17059 (2011).

(8) Dieudonné, B. et al. The key role of the composition and structural features in fluoride ion conductivity in tysonite $Ce_{1-x}Sr_xF_{3-x}$ solid solutions. *Dalton Trans.* **46**, 3761–3769 (2017).

(9) Dieudonné, B. et al. Exploring the $Sm_{1-x}Ca_xF_{3-x}$ tysonite solid solution as a solid-state electrolyte: relationships between structural features and F^- ionic conductivity. *J. Phys. Chem. C.* **119**, 25170–25179 (2015).

(10) Preishuber-Pflügl, F., Epp, V., Nakhal, S., Lerch, M. & Wilkening, M. Defect-enhanced F^- ion conductivity in layer-structured nanocrystalline BaSnF4 prepared by high-energy ball milling combined with soft annealing. *Phys. Status Solidi C.* **12**, 10–14 (2015).

(11) Ahmad, M. M., Yamane, Y. & Yamada, K. Structure, ionic conduction, and giant dielectric properties of mechanochemically synthesized $BaSnF_4$. *J. Appl. Phys.* **106**, 074106 (2009).

(12) Sorokin, N. I. SnF_2 -Based Solid Electrolytes. *Inorg. Mater.* **40**, 989–997 (2004).

(13) Mohammad, I., Witter, R., Fichtner, M. & Anji Reddy, M. Room-temperature, rechargeable solid-state fluoride-ion batteries. *ACS Appl. Energy Mater.* **1**, 4766–4775 (2018).

(14) McRae, L. et al. Design of semiconducting electrides via electron-metal hybridization: the case of Sc_2C. https://doi.org/10.33774/chemrxiv-2021-t1b6t (2021).

(15) Druffel, D. L. et al. Synthesis and electronic structure of a 3D crystalline stack of mxene-like sheets. *Chem. Mater.* **31**, 9788–9796 (2019).

(16) Warren, S. C. et al. Metal carbides and metal nitrides for a fluoride ion battery. *US Patent App.* US20210151755A1, 1–49 (2021).

(17) Shang, S.-L. et al. A comprehensive first-principles study of pure elements: vacancy formation and migration energies and self-diffusion coefficients. *Acta Mater.* **109**, 128–141 (2016).

(18) Zhou, B.-C., Shang, S.-L., Wang, Y. & Liu, Z.-K. Diffusion coefficients of alloying elements in dilute Mg alloys: A comprehensive first-principles study. *Acta Mater.* **103**, 573–586 (2016).

(19) Angsten, T., Mayeshiba, T., Wu, H. & Morgan, D. Elemental vacancy diffusion database from high-throughput first-principles calculations for fcc and hcp structures. *N. J. Phys.* **16**, 015018 (2014).

(20) Zeng, Y., Li, Q. & Bai, K. Prediction of interstitial diffusion activation energies of nitrogen, oxygen, boron and carbon in bcc, fcc, and hcp metals using machine learning. *Comp. Mater. Sci.* **144**, 232–247 (2018).

(21) Wu, M. et al. High-throughput screening of TMOCl cathode materials based on the full-cell system for chloride-ion batteries. *J. Mater. Chem. A* **9**, 23169–23177 (2021).

(22) Feng, Y. et al. High-throughput modeling of atomic diffusion migration energy barrier of fcc metals. *Prog. Nat. Sci.: Mater. Int.* **29**, 341–348 (2019).

(23) Wu, H., Mayeshiba, T. & Morgan, D. High-throughput ab-initio dilute solute diffusion database. *Sci. Data* **3**, 160054 (2016).

(24) Kim, K. et al. Material Design Strategy for Halide Solid Electrolytes Li_3MX_6 (X = Cl, Br, and I) for All-Solid-State High-Voltage Li-Ion Batteries. *Chem. Mater.* **33**, 3669–3677 (2021).

(25) Zhong, W. & Zhao, J.-C. A comprehensive diffusion mobility database comprising 23 elements for magnesium alloys. *Acta Mater.* **201**, 191–208 (2020).

(26) Agarwal, R. & Trinkle, D. R. Ab initio magnesium-solute transport database using exact diffusion theory. *Acta Mater.* **150**, 339–350 (2018).

(27) He, B. et al. High-throughput screening platform for solid electrolytes combining hierarchical ion-transport prediction algorithms. *Sci. Data* **7**, 151 (2020).

(28) Kahle, L., Marcolongo, A. & Marzari, N. High-throughput computational screening for solid-state Li-ion conductors. *Energy Environ. Sci.* **13**, 928–948 (2020).

(29) Morkhova, Y. A. et al. Computational search for novel Zn-Ion conductors—a crystallochemical, bond valence, and density functional study. *J. Phys. Chem. C.* **125**, 17590–17599 (2021).

(30) Sendek, A. D. et al. Holistic computational structure screening of more than 12000 candidates for solid lithium-ion conductor materials. *Energy Environ. Sci.* **10**, 306–320 (2017).

(31) Pizzi, G., Cepellotti, A., Sabatini, R., Marzari, N. & Kozinsky, B. AiiDA: automated interactive infrastructure and database for computational science. *Comp. Mater. Sci.* **111**, 218–230 (2016).

(32) Mathew, K. et al. Atomate: a high-level interface to generate, execute, and analyze computational materials science workflows. *Comp. Mater. Sci.* **139**, 140–152 (2017).

(33) Jain, A. et al. FireWorks: a dynamic workflow system designed for high-throughput applications. *Concurrency Computat.: Pract. Exp.* **27**, 5037–5059 (2015).

(34) Curtarolo, S. et al. AFLOW: an automatic framework for high-throughput materials discovery. *Comp. Mater. Sci.* **58**, 218–226 (2012).

(35) Saal, J. E., Kirklin, S., Aykol, M., Meredig, B. & Wolverton, C. Materials design and discovery with high-throughput density functional theory: the open quantum materials database (OQMD). *JOM* **65**, 1501–1509 (2013).

(36) Zapata, F. et al. Qmflows: A tool kit for interoperable parallel workflows in quantum chemistry. *J. Chem. Inf. Model.* **59**, 3191–3197 (2019).

(37) Adorf, C. S., Dodd, P. M., Ramasubramani, V. & Glotzer, S. C. Simple data and workflow management with the signac framework. *Comp. Mater. Sci.* **146**, 220–229 (2018).

(38) Mayeshiba, T. et al. The MAterials Simulation Toolkit (MAST) for atomistic modeling of defects and diffusion. *Comp. Mater. Sci.* **126**, 90–102 (2017).

(39) Sundberg, J. D. Online Github repository for Simmate. https://github.com/jacksund/simmate (2022).

(40) Jain, A. et al. Commentary: The Materials Project: A materials genome approach to accelerating materials innovation. *APL Mater.* **1**, 011002 (2013).

(41) Ong, S. P. et al. Python Materials Genomics (pymatgen): A robust, open-source python library for materials analysis. *Comp. Mater. Sci.* **68**, 314–319 (2013).

(42) Smidstrup, S., Pedersen, A., Stokbro, K. & Jónsson, H. Improved initial guess for minimum energy path calculations. *J. Chem. Phys.* **140**, 214106 (2014).

(43) Trottier, R. M., Millican, S. L. & Musgrave, C. B. Modified single iteration synchronous-transit approach to bound diffusion barriers for solid-state reactions. *J. Chem. Theory Comput.* **16**, 5912–5922 (2020).

(44) Henkelman, G., Uberuaga, B. P. & Jónsson, H. A climbing image nudged elastic band method for finding saddle points and minimum energy paths. *J. Chem. Phys.* **113**, 9901–9904 (2000).

(45) Rong, Z., Kitchaev, D., Canepa, P., Huang, W. & Ceder, G. An efficient algorithm for finding the minimum energy path for cation migration in ionic materials. *J. Chem. Phys.* **145**, 074112 (2016).

(46) Garrido Torres, J. A., Jennings, P. C., Hansen, M. H., Boes, J. R. & Bligaard, T. Low-scaling algorithm for nudged elastic band calculations using a surrogate machine learning model. *Phys. Rev. Lett.* **122**, 156001 (2019).

(47) Sheppard, D., Terrell, R. & Henkelman, G. Optimization methods for finding minimum energy paths. *J. Chem. Phys.* **128**, 134106 (2008).

(48) Peterson, A. A. Acceleration of saddle-point searches with machine learning. *J. Chem. Phys.* **145**, 074106 (2016).

(49) Li, H.-X., Zhou, X.-Y., Wang, Y.-C. & Jiang, H. Theoretical study of Na^+ transport in the solid-state electrolyte Na_3OBr based on deep potential molecular dynamics. *Inorg. Chem. Front.* **8**, 425–432 (2021).

(50) Chen, H., Wong, L. L. & Adams, S. SoftBV - a software tool for screening the materials genome of inorganic fast ion conductors. *Acta Crystallogr. B: Struct. Sci. Cryst. Eng. Mater.* **75**, 18–33 (2019).

(51) Wang, Y. et al. Design principles for solid-state lithium superionic conductors. *Nat. Mater.* **14**, 1026–1031 (2015).

(52) Zhong, J., Chen, L. & Zhang, L. Automation of diffusion database development in multicomponent alloys from large number of experimental composition profiles. *NPJ Comput. Mater.* **7**, 35 (2021).

(53) Zhang, L. et al. A database of ionic transport characteristics for over 29000 inorganic compounds. *Adv. Funct. Mater.* 30, 2003087 (2020).

(54) Elbaz, Y. & Furman, D. & Caspary Toroker, M. Modeling diffusion in functional materials: from density functional theory to artificial intelligence. *Adv. Funct. Mater.* **30**, 1900778 (2020).

(55) Danto, Y., Poujade, G., Pistré, J. D., Lucat, C. & Salardenne, J. A Pb|PbF_2 | BiF_3|Bi thin solid film reversible galvanic cell. *Thin Solid Films* **55**, 347–354 (1978).

(56) Mohammad, I., Witter, R., Fichtner, M. & Reddy, M. A. Introducing interlayer electrolytes: toward room-temperature high-potential solid-state rechargeable fluoride ion batteries. *ACS Appl. Energy Mater.* **2**, 1553–1562 (2019).

(57) Berastegui, P. Structure and conductivity of some fluoride ion conductors. *Solid State Ion.* **154–155**, 605–608 (2002).

(58) Wapenaar, K. E. D. The ionic conductivity of fluorite-structured solid solutions of composition: MF_2:UF_4:CeF_3 (M = Ca, Sr, Ba). *J. Electrochem. Soc.* **126**, 667 (1979).

(59) Kennedy, J. H. Ionic conductivity of doped beta-lead fluoride. *J. Electrochem. Soc.* **123**, 47 (1976).

(60) Anji Reddy, M. & Fichtner, M. Fluoride-ion conductors. *Handb. Sol. State Batteries* **6**, 277–306 (2015).

(61) Nowroozi, M. A., Wissel, K., Rohrer, J., Munnangi, A. R. & Clemens, O. $LaSrMnO_4$: reversible electrochemical intercalation of fluoride ions in the context of fluoride ion batteries. *Chem. Mater.* **29**, 3441–3453 (2017).

(62) Nowroozi, M. A., de Laune, B. & Clemens, O. Reversible electrochemical intercalation and deintercalation of fluoride ions into host lattices with schafarzikite-type structure. *Chem. Open* **7**, 617–623 (2018).

(63) Clemens, O. et al. Electrochemical fluorination of perovskite type $BaFeO_{2.5}$. *Dalton Trans.* **43**, 15771–15778 (2014).

(64) Deng, Z., Zhu, Z., Chu, I.-H. & Ong, S. P. Data-driven first-principles methods for the study and design of alkali superionic conductors. *Chem. Mater.* **29**, 281–288 (2017).

(65) Ward, L. et al. Matminer: An open source toolkit for materials data mining. *Comp. Mater. Sci.* **152**, 60–69 (2018).

(66) Gražulis, S. et al. Crystallography open database - an open-access collection of crystal structures. *J. Appl. Crystallogr*. **42**, 726–729 (2009).

(67) Yuliia, P., Anatoliy, O. & Anton, N. Synthesis and electrical conductivity of solid solutions of the system PbF_2-NdF_3-SnF_2. *Ukr. Chem. J.* **86**, 24–37 (2020).

(68) Sorokin, N. I. Anion-conducting fluoride and oxyfluoride glasses. *Russ. Chem. Rev.* **70**, 801–807 (2001).

(69) Choudhury, P., Mandal, P., Das, A. N. & Ghosh, B. Dielectric behaviour and thermal expansions of $ZnTiF_6$ $6H_2O/6D_2O$ and $MnTiF_6$ $6H_2O/6D_2O$ crystals. *J. Phys. C: Solid State Phys.* **19**, 3961–3970 (1986).

(70) Rubins, R. S. & Yung, Y. H. Pair and isolated ion spectra of Ni^{2+} in $MgTiF_6$ $6H_2O$. *J. Chem. Phys.* **75**, 4285–4288 (1981).

(71) Kawamoto, Y. Ionic conductivities of ZrF_4-BaF_2-CsF glasses. *Solid State Ion.* **22**, 207–212 (1987).

(72) Kresse, G. & Furthmüller, J. Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Phys. Rev. B* **54**, 11169–11186 (1996).

(73) Lenstra, A. K., Lenstra, H. W. & Lovász, L. Factoring polynomials with rational coefficients. *Math. Ann.* **261**, 515–534 (1982).

(74) Larsen, P. M., Pandey, M., Strange, M. & Jacobsen, K. W. Definition of a scoring parameter to identify low-dimensional materials components. *Phys. Rev. Materials* **3**, 034003 (2019).

(75) Pan, H. et al. Benchmarking coordination number prediction algorithms on inorganic crystal structures. (2020) doi:10.26434/chemrxiv.12508229.v1.

(76) Mehl, M. J. et al. The AFLOW library of crystallographic prototypes: part 1. *Comp. Mater. Sci.* **136**, S1–S828 (2017).

(77) Hicks, D. et al. The AFLOW library of crystallographic prototypes: part 2. *Comp. Mater. Sci.* **161**, S1–S1011 (2019).

(78) Hicks, D. et al. The AFLOW library of crystallographic prototypes: part 3. *Comp. Mater. Sci.* **199**, 110450 (2021).

(79) Prices of chemical elements - Wikipedia. https://en.wikipedia.org/wiki/Prices_of_chemical_elements/.

(80) Wolfram | Alpha Examples: Materials. https://www.wolframalpha.com/.

CHAPTER 5: HIGH-THROUGHPUT DISCOVERY OF LAYERED ELECTRIDES FROM FLUORIDE ANALOG SYSTEMS

5.1 Summary

Electrides possess exceptional electronic properties due to lone electrons in distinct lattice sites, where their lone electrons behave as anions would in traditional ionic materials. Despite their fascinating properties, few inorganic electrides have been experimentally realized because they are primarily explored through high-temperature bulk synthesis, which typically yields only the most stable polymorphs. Here, we propose an alternative low-temperature synthetic strategy for electrides via the etching of halide materials. To identify candidate materials for this synthetic approach, we outline a search strategy for the high-throughput discovery of novel electrides from all known materials, and we illustrate this approach using a smaller dataset of layered fluoride-ion conductors. This resulted in several candidate electrides and structure types that can be explored via low-temperature defluorination and, more importantly, confirmed that many more electrides are accessible as kinetically trapped phases rather than as thermodynamic ground states. Our search results support our strategy for the dehalogenation of known materials, which can give rise to a broader class of electride materials. These previously inaccessible materials will advance our understanding of the divergent properties of electrides and facilitate their applications as highly reducing and conducting systems.

5.2 Introduction

Traditional ionic solids are composed entirely of cation and anions, and all electrons are associated with individual atoms. However, in rare cases, ionic solids can diverge from our fundamental chemical rules and possess electrons that are not associated with any atomic orbital – i.e., lone electrons that occupy distinct lattice sites and behave as anions.[1] These materials are known as electrides.

The anionic electrons in electrides give rise to exceptional electronic properties, which have attracted considerable attention in recent literature.[1,2,3] While traditional ionic solids are frequently hard, brittle, and electrically insulating, electrides can be soft, malleable, and among the most conductive materials known. To this end, a single electride material (Ca_2N, which can also be written as $[Ca_2N]^+\cdot e^-$) has demonstrated conductivities competitive with silver[4], high catalytic activity in challenging reactions[5,6], and electron donating effects that exceed alkali metals[7]. In addition to serving as superconductors, cocatalysts, and solid-state dopants, these electron-rich materials have more recently been proposed as high-voltage, high-capacity anodes for anion-shuttle batteries.[8] Surprisingly, an investigation of anion shuttling proposed that electride character fundamentally alters the mechanism and redox process by which anions are (de)intercalated.[8] Thus, electrides have proven to be an exciting class of materials with diverse applications, and further, their unique characteristics are continuing to reshape and challenge our chemical understanding.

Unfortunately, however, there are only a limited number of synthesizable electrides and, as a result, the chemical boundaries that define where one is likely to find electrides remain unknown.[2] Candidate electrides have been proposed from high-throughput searches of all existing compounds[9,10,11,12], but these materials frequently possess minimal or no electride character after further investigation. Electride electrons are highly reducing and high energy, and it is therefore no surprise that a relatively small number of thermodynamically stable electrides exist. Moreover, the majority of known electrides fall within a few structure types where the electride character is stable.[2] Because of this, elemental substitution of known electrides has been more successful in realizing new electride compounds relative to high-throughput computational searches.[13,14] Ultimately, experimentalists need a more effective strategy to identify new electrides and understand their properties.

Because electrides are frequently observed in chemically strained systems (induced by incommensurate stoichiometries or even extreme mechanical pressures)[15,16,17], we propose that the majority of electride materials exist as metastable phases. Low-temperature syntheses should therefore be explored to avoid decomposition of these chemically strained systems. Inspired by recent reports (de)intercalation of layered electrides[8] and on fluoride mobility[18], we propose that low-temperature removal of halides will result in previously inaccessible electride materials. Thus, this work outlines a generalizable strategy for electride synthesis and proposes several candidate materials and structure types to investigate via this approach. Our findings suggest that dehalogenation can produce the first metastable electride materials and provide novel electrides that further our chemical understanding.

5.3 Design strategy for novel electrides

In the past, the synthesis of inorganic electrides has been exclusively carried out via high-temperature reactions, which limits known electrides to systems that are thermodynamically favored.[1,2,3,13,14] However, even in these cases, the synthesis of electrides can be extremely challenging due to the similar energy of competing phases. Therefore, an alternative strategy is needed that would kinetically favor the desired phase.

In contrast to high-energy electrides, inorganic halides are often dynamically stable and easier to synthesize as pure phases.[19,20] For example, the bulk synthesis of a layered electride $[Y_2C]^{2+}\cdot(e^-)_2$ is highly sensitive to reaction parameters and frequently contains mixed phases, whereas its fluoride-substituted analog (Y_2CF_2) gives highly crystalline phases and is less susceptible to oxidation.[21] With this in mind, we also emphasize that lone electride electrons (e^-), hydrides (H-), and halides ($X=F^-$, Cl^-, Br^-, I^-) can be viewed as likely substitutions due to their similar oxidation states (e.g., e^- for X^- and vice versa) (Figure 1-1). For example, an inorganic fluoride can be etched to produce an electride according to the half-reaction:

$$[Y_2C]^{2+}(F^-)_2 + 2\,e^- \rightarrow [Y_2C]^{2+}(e^-)_2 + 2\,F^-$$

Analogous half reactions can be written for hydride and other halides as well. We can generalize the base electride (Y_2C) to produce a more general half reaction, where X is any halide and E is the base lattice for any electride/halide:

$$[E]^{2+}(X^-)_n + n\,e^- \rightarrow [E]^{2+}(e^-)_n + n\,X^-$$

Considering the prevalence of inorganic halides, this model suggests an alternative synthetic approach for electrides: the low-temperature dehalogenation of the electride’s halide analog. Thus, we propose electrochemical or chemical dehalogenation to form new, previously inaccessible electrides. This can also be a preferable synthetic strategy relative to the high-temperature melts often used for known inorganic electrides. Here, we focus on the discovery of novel electrides to expand our understanding. We therefore sought to identify known halide materials that are promising for dehalogenation.

For candidate materials, either an electrochemical bias or chemical etching strategy must be thermodynamically and kinetically favorable at low temperature. To this end, we have recently predicted that electrides can undergo reversible intercalation of fluoride at room temperature, which suggests our strategy is indeed viable for known electrides such as Y_2C.[8] We therefore propose that fluoride mobilities and reduction potentials can provide insight into whether electrochemical fluorination of other halide materials is possible – even at slightly elevated temperatures (e.g., <200°C). For chemical etching, we also propose that halide materials can be converted using highly reducing alkali metals such as lithium:

$$[E]^{2+}(X^-)_n + n\,Li \rightarrow [E]^{2+}(e^-)_n + n\,LiX$$

We note that in addition to lithium metal, other reducing metals can be substituted. Regardless of the reducing agent, this process will require good halide transport and likely require the reducing agent to be in a liquid or vapor phase at low temperatures. Overall, both electrochemical and alkali-metal strategies should be evaluated when looking at potential halide systems, so this study investigates both routes for all candidate materials.

5.4 Selecting candidate materials

Evaluating all known halide materials would be an ambitious and computationally expensive task. Even for a single metric such as fluoride mobility, standard methods can require hundreds of CPU hours for a single compound[22,23], and further, follow-up analyses that evaluate dynamic or thermodynamic stabilities simply add to this challenge. We must therefore limit this study to a subset of model material systems.

Fortunately, we have previously explored all known fluoride materials to identify those with high fluoride mobility – a central requirement to our proposed synthetic strategy.[18] This ultimately led to fluoride mobility predictions across ~7,000 ionic pathways, and many of these fluoride-ion conductors could be promising candidates for low-temperature fluoride removal. In fact, our dataset includes fluoride analogs for well-known electride structure types, such as Ho_2CF_2, which is isostructural with other M_2C (M=Sc, Y, Gd, Tb, Dy, Ho) and M_2N (M=Ca, Sr, Ba) electrides. We therefore use this dataset of ionic mobilities to limit our search for low-temperature dehalogenation candidates.

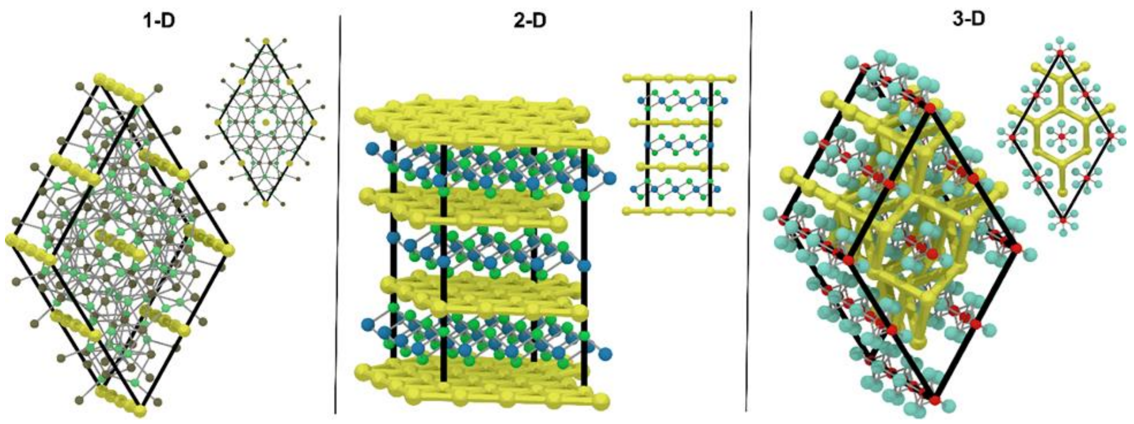


Figure 5-1. The dimensionality of fluoride-electride networks. A network's dimensionality is based on the connectivity of the anionic electron lattice positions. Shown are superlattices of the 1-D electride La_5Pb_3 (left), 2-D electride Ca_2N (middle), and 3-D electride Cs_3O (right), where the electride network is in yellow. Insets are alternative perspectives of the superlattice. It is important to note that Cs_3O is likely not a 3-D electride in reality, as each anionic electron site is largely isolated (>6 Å separation) and e- exchange between positions would be extremely rare. We still show it as such to serve as an example because no 3-D electrides have been reported.

In addition to fluoride being mobile, the removal of fluoride from a host lattice also requires a percolation pathway for fluoride to diffuse along. We refer to this pathway as the dimensionality of the fluoride (and electride) network (Figure 5-1).[24] Historically, layered (2D) electrides have been the most widely used in applications such as electron donors and battery electrodes. Most notably, a single 2D structure type (rocksalt MoS_2-type) has also been the focus of the majority of inorganic electride research.[2] We seek to focus this study on the discovery of new 2D electrides to build upon current understandings and technologies.

To complete our input criteria, we also limit our search to materials that contain at least 2 elements other than fluoride and that have good thermodynamic stability. The former criterion excludes metal fluorides (MF_x), while the latter ensures that our starting fluoride is indeed synthesizable. Together, we arrive at the following filtering criteria for our fluoride-mobility dataset given in Sundberg et al.[18]: (i) the host lattice dimensionality is 2D, (ii) the fluoride/electride percolation network is 2D, (iii) the fluoride structure contains at least three elements, (iv) there is a fluoride diffusion pathway with a barrier below 1 eV, and (v) the hull energy (E_{hull}) is below 100 meV. This resulted in 233 fluoride materials that served as the starting candidate materials for this study.

While we limit our search to layered fluorides, we emphasize that both our previous study and methods used here are applicable to other percolation types (1D and 3D) and other hydrides/halides (H^-, Cl^-, Br^-, I^-). This study thus serves as a model for the exploration of other systems and the realization of novel electride materials.

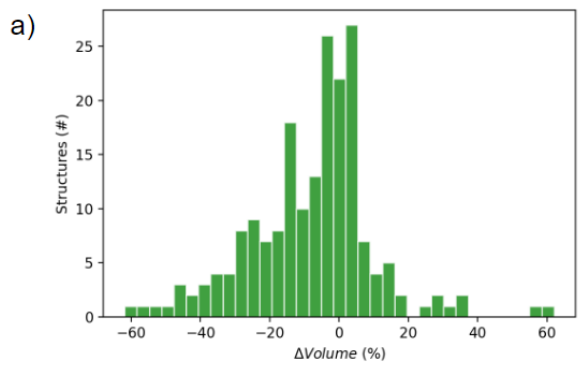
5.5 Characterization of candidate materials

5.5.1 Structure optimization

The 233 layered fluorides were then used to generate potential electrides. This was done by removing all fluoride species from each structure and then geometrically optimizing the result. Relaxations were run using Materials Project settings[25] through the Simmate workflow framework[26]. To quickly identify which structures were particularly interesting (or problematic), we evaluated the relaxation results for excessive changes in volume.

Electride electrons can vary in their effective size relative to other -1 anions, which indicates that the removal of fluoride may lead to expansion or contraction of the layered structures' interlayer.[2] This can be seen in two example systems, Ca_2NF and Y_2CF_2. In the former, removal of fluoride causes the lattice volume to change by +12.3%, while in the latter, fluoride removal results in a -7.7% volume change. Despite having isostructural host lattices, an electride electron's effective size can result in different lattice changes. In some cases, however, instability of the host lattice structure would result in extremely large compression of the lattice and even result in atomic rearrangement upon defluorination. This is particularly important to consider because we removed fluoride from structures whose host lattices have not yet been analyzed for dynamic stability. Thus, the relative volume change during relaxations can be used to identify structures unlikely to exist as electride materials.

The distribution of volume changes for all relaxations is shown in Figure 5-2a, where 143 (of 194 completed relaxations) structures change less than ±20%. This is consistent with many layered electrodes that experimentally expand/contract by ±5-15%.[28] Larger volume changes are possible in certain instances, but are usually accompanied by significant decomposition.[28,29] Therefore, we consider modest volume changes (< ±20%) to indicate the most promising candidates for the low-temperature formation of stable electride materials.



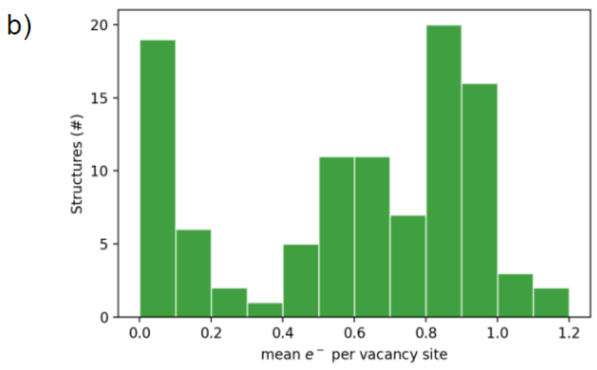


Figure 5-2. Distribution of volume changes and electride characteristics upon defluorination. Fluoride was removed from all candidate structures, and the resulting structure was relaxed. (a) The distribution of volume changes (relative to the original fluoride structures) illustrates the variable effective size of electride electrons and the collapse/rearrangement of the host lattice. (b) Structures that undergo smaller changes were evaluated via Bader-ELF analysis by placing empty atoms at previously occupied fluoride sites, and the mean electron density for each overall structure is given.

5.5.2 Structure matching

All relaxed structures were also matched to known phases/systems in the Materials Project database[25] to identify which electride candidate materials were previously synthesized. Structure matching was performed using (i) exact symmetry matching and (ii) fingerprint similarity matching.

For (i), the structure matcher implemented in PyMatGen was used with default symmetry tolerances, which are very loose (vector lengths = 0.2 Å, vector angles = 5°, and site coordinates = 0.3 Å) to capture slight distortions.[30] This resulted in 12 of the 233 structures being matched to known phases in the database. 11 of these 12 structures are fluorides of the Matlockite structure type and are primarily metal sulfides (Figure 5-3a). The matching structures were actually of the NaCl-rocksalt structure type, which is a slight distortion of the Matlockite host lattice. This suggests that when fluoride is removed from these materials, the host lattices will likely collapse across the interlayer spacing. We performed short molecular dynamics simulations on these defluorinated materials at 1000 K and, as expected, the structures collapsed

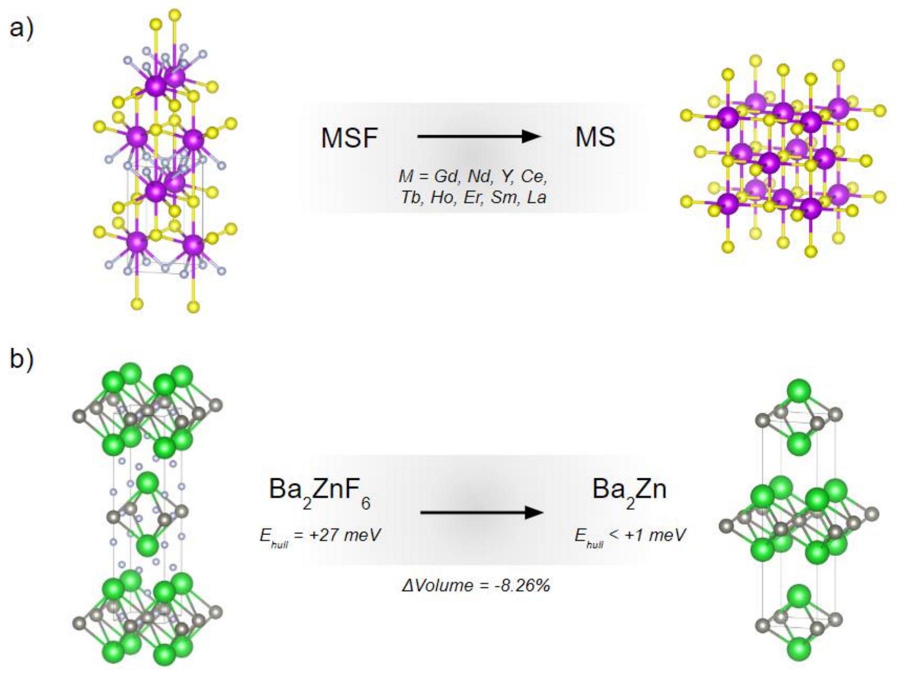


Figure 5-3. Defluorinated structures matched to experimentally known phases. (a) Matlockite fluorides (left) were matched to rocksalt metal sulfides (right), indicating that the MS host lattice likely collapses upon defluorination. However, this matching does suggest that the rocksalt crystals may form electrides under high pressure. (b) Ba_2ZnF_6 defluorination occurs in both the inter- and intralayer, where a surprisingly small change in volume is observed after ~66% loss in atomic density.

to rocksalt structures within 1 picosecond. These metal sulfides are therefore incompatible with our proposed low-temperature synthetic strategies. However, these rocksalt phases may still form electrides under extreme pressure (as is done with other electron-rich materials such as sodium metal). We therefore suggest that this series of rocksalt metal sulfides be investigated under high-pressure environments.

The extra structure matched was the Ba_2Zn-Ba_2ZnF_6 electride-halide analog (Figure 5-3b), which undergoes a surprisingly small volume change of -8.26% during defluorination and the electride Ba_2Zn phase <1 meV above the hull. Fluoride intercalation unexpectedly occurs in both the inter- and intra-layer of the Ba_2Zn layers. We therefore highlight Ba_2Zn as a material of interest for electride research due to its incommensurate, electron-rich stoichiometry of Ba vs. Zn and unusually high threshold for anion intercalation (200% atomic capacity).

For structure matching via fingerprint similarities (ii), we applied the CrystalNN fingerprint[31] through the MatMiner code[32]. This fingerprint allows us to capture structures with similar coordinates and structure types and, as is particularly important for layered materials, capture matches that have different stacking sequences. To our surprise, only one new structure match is found (compared to exact structure matching). A match was found for Ho_2C-Ho_2CF_2, which is already an experimentally known electride.[33] This finding confirms that defluorination does indeed produce electride materials, but the lack of other matches suggests that nearly all of our candidate materials are novel compounds.

5.5.3 Thermodynamic stability

We then performed database searches of the electride chemical systems to evaluate which ones were previously explored and to identify competing phases. 28 of the 233 electride analogs (12%) did not have any known structures in the same chemical system – i.e., the candidate material $BaSrPdO_2$ would have no known phases in the $Ba_wSr_xPd_yO_z$ compositional space (where variables w, x, y, and z are >0). These candidate compounds correspond to chemical systems that are poorly explored, and full phase searches would be required to understand the thermodynamic plausibility of such structures. Additionally, 149 out of 233 electride analogs had structures with matching reduced compositions in the database. However, because exact structure matching only gave matches for 14 structures (see discussion above), the remaining 135 structures are likely metastable phases. These searches indicate that the large majority of systems are unexplored phases or compositions and that more robust thermodynamic analysis is required to evaluate the stability of our candidate electrides.

In the simplest approach, we can compare the thermodynamic stability of our candidate electrides to known materials in the database. Simmate uses updated pseudopotentials relative to the Material Project database, so we recalculated known stable phases for the full chemical system of our candidate electride phases. These several hundred materials were reoptimized using the Simmate workflow engine, and then calculated enthalpies were used to predict the hull energy (E_{hull}) for each candidate electride. From these results, only two structures ($Li_2V_2O_4$ and Na_4Pd_2) are predicted to be on the hull, while 32 of 233 have E_{hull} < 100 meV. This is consistent with our predictions from structure matching and systems search, which suggested that nearly all candidate electrides are metastable. However, we reemphasize that metastable electrides are in fact the expected outcome of this search strategy. This agrees with our expectation that the majority of existing electrides are metastable due their electron-rich and chemically strained compositions, which reinforces the need for a low-temperature synthetic strategy to experimentally realize novel electrides.

5.5.4 Low-temperature defluorination

Above, we proposed the low-temperature synthesis of electrides driven by an electrochemical bias or a strong reducing agent. Here, we evaluate possible reactions by predicting (i) deintercalation potentials for fluoride and (ii) the reaction enthalpies to reduce each candidate fluoride.

To approximate the intercalation potential of inserting an electride electron into an inorganic fluoride (vs. Li/Li$^+$), we adopt the method reported by Druffel et al.[8] and provide an overview of this approach using Y_2CF_2 as an illustrative example. Because the solvation energy of fluoride cannot be easily calculated, we approximate this term using the ionization energy of lithium metal and solvation energy of lithium fluoride:

$$
\begin{array}{lr}
Y_2CF_{2(s)} + 2\,Li_{(s)} \rightarrow [Y_2C]^{2+}(e^-)_{2(s)} + \ 2\,LiF_{(s)} & \mathbf{\Delta G_{rxn}} \\
2\,LiF_{(s)} \rightarrow 2\,Li^+_{(solvated)} + 2\,F^-_{(solvated)} & \mathbf{\Delta G_{solv}} \\
+ \quad 2\,Li^+_{(solvated)} + 2\,e^- \rightarrow 2\,Li_{(s)} & \mathbf{E^{Li^+}_{red}} \\
\hline
Y_2CF_{2(s)} + 2\,e^- \rightarrow [Y_2C]^{2+}(e^-)_{2(s)} + \ 2\,F^-_{(solvated)} & \mathbf{E^{Y_2CF_2}_{e^-int}}
\end{array}
$$

ΔG_{rxn} is approximated using the enthalpy of the solid reaction via DFT, neglecting the change in entropy. ΔG_{solv} uses an experimental value of the energy of solvation of LiF (21.13 kJ/mol) in the ionic liquid, 1-butyl-3-methylimidazolium trifluoromethanesulfonate ([C_4C_1im][OTf]). Through the Nernst equation, we can relate the Gibbs free energy of the reduction reaction to the voltage of the cell vs. Li/Li$^+$. This gives the relation:

$$E^{Y_2CF_2}_{e^-int_{vs.\ Li/Li^+}} = \frac{\Delta G_{rxn} + \ \Delta G_{solv}}{-nF}$$

where *F* is the Faraday constant and *n* is the number of electrons transferred. Thus, we can predict an approximate intercalation potential by calculating the ΔG_{rxn} of the fluoride analog (e.g., Y_2CF_2) with Li metal to form the electride analog and LiF (repeating the equation from above):

$$Y_2CF_{2(s)} + 2\,Li_{(s)} \rightarrow [Y_2C]^{2+}(e^-)_{2(s)} + 2\,LiF_{(s)} \qquad \mathbf{\Delta G_{rxn}}$$

Coincidentally, this is also one possible reaction to chemically etch our candidate fluoride structures with a strong reducing agent (i.e., Li metal). We therefore use this general reaction to evaluate all of our candidate materials for both electrochemical and chemical removal of fluoride:

$$[E]^{2+}(F^-)_n + n\,Li \rightarrow [E]^{2+}(e^-)_n + n\,LiF$$

This calculation resulted in 171 of the 233 fluorides with a favorable (negative) ΔG_{rxn} with Li metal, indicating that in many cases this would be a viable synthetic route. We do note, however, that this analysis neglects electride stability as well as the possibility of using a stronger reducing agent than Li metal. Our findings from other analyses are therefore higher priority when identifying the most promising candidate materials, and in some cases, a stronger reducing agent should be considered. We therefore provide theoretical E_{interc} and ΔG_{rxn} not as a required criterion but instead as a guide for possible experimental investigation.

5.5.6 Electride character and electronic structure

Hitherto, we have predicted the thermodynamic stabilities and synthetic strategies for candidate materials, but we have not yet confirmed that these materials are in fact electrides after fluoride removal. To do this, oxidation states and electride character can be evaluated via several techniques, but it is important to consider how the presence of lone (and sometimes diffuse) electride electrons complicates population analyses.[34,35]

The most commonly applied strategy is to visualize the electron localization function (ELF).[35,36] The ELF corresponds to the probability of finding an electron of the same spin in a region of space. Thus, smaller probabilities correspond to a more localized electron. Furthermore, recall electrides are characterized by high electron density at lattice sites where there are no atomic species. Other high-throughput searches for electrides have sought to quantify ELF by drawing an ELF isosurface and then integrating charge densities within these confined regions.[9] Unfortunately, however, localization probabilities vary greatly between crystal structures, and this process can frequently give inconsistent results across different compositions and structure types.

Meanwhile, for traditional ionic solids, oxidation states are frequently determined by Bader analysis, where charge density is partitioned in a consistent manner using zero-flux surfaces of the charge density.[37] While extremely robust for most ionic solids, Bader analysis struggles to evaluate electride materials due to the low total electron density in electride regions (relative to atomic sites). This ultimately leads to underestimation of the electride character, where the electron density is incorrectly assigned to one of the neighboring atoms.

We therefore introduced a new partitioning strategy that addresses the shortcomings of Bader analysis and fixed-cutoff ELF isosurfaces. Specifically, we use the Bader algorithm to find zero-flux surfaces in the ELF in a consistent manner, and then these surfaces are used to partition the charge density. We found that this approach effectively captures expected oxidation states in both common ionic materials and electrides (see Section 2.3.4).

We expect that electride electrons localize in the same lattice sites as fluoride from the original structure. Therefore, using the Bader/ELF method and fluoride materials as templates, we introduce "empty" atoms where electride electrons would be expected and use these sites to quantify the electride character. This process was automated and applied to all candidate materials, where 109 of 233 materials completed successfully. Workflow failures were caused either by failed convergence of the Bader algorithm or by the host lattice distorting during relaxation, which led to our algorithm's inability to match electride-halide pairings and insert empty atoms.

As expected, the results show that the majority of materials possess 0-1 electrons at expected electride sites. The distribution of mean electron density per site (Figure 5-2b) shows two clear groups of materials, centered at 0e^-/site and 0.9e^-/site. We interpret this to be the separation of materials that do and do not form electrides, respectively. Only 17 of the 109 materials contained zero electride character (0e^-/site), whereas the large majority (75 total, 69%) of defluorinated structures contained >0.5e^-/site. We also note that several compounds contain >2 e^-/site, but after closer inspection of these materials, it was found that the extremely high electride characteristics are an artifact of misassigning atomic charge densities to electride sites. This occurred in materials where volume changes were in excess of -40%, resulting in empty atoms being placed too close (<1 Å) to atomic sites. These results indicate that etching (i.e., removal or deintercalation) of F^- species from known inorganic fluorides frequently leads to electride formation.

5.6 Exploring the final dataset

Finally, we analyze the results from section 5.5 to identify the most promising candidates for defluorination. We filter down to electride materials that (i) are within 100 meV of the hull, (ii) undergo volume changes of $< \pm 10\%$, and (iii) have > 0.2 e^- per vacancy site. This results in eight final materials that can be considered for higher quality calculations.

Of these eight materials, the most notable entry is the Ho_2C-Ho_2CF_2 system, which is also the only previously known electride expected from our search strategy. This indicates that our final criteria are successful at isolating promising electride candidates. Furthermore, the lack of other isostructural electrides (e.g., Y_2C) highlights how this study is limited by known fluorides in the Materials Project database. Thus, there is a need for a more general search of promising structure types and prediction of new fluoride materials.

Three crystals of these eight materials are derived from Matlockite fluoride structures, which we found are likely to collapse to rocksalt crystals upon defluorination (see discussion above). Therefore, these three structures (NdTe, BaBr, and EuCl) should instead be evaluated for electride characteristics under extreme pressure rather than low-temperature defluorination.

The remaining four materials are our most promising candidates for low-temperature defluorination (Figure 5-4). Two of these are isostructural oxides (YOF and LaOF), and others are BaAlGeF and $Sr_2Ti_2As_2OF_2$. All materials show electride character upon defluorination but to varying degrees (0.47 to 0.82 e^- per site). This constitutes a broad range of electride character that has not been previously observed. Historically, definitions of electrides and traditional ionic materials have been categorical, but our recent findings show that electride electrons can hybridize with nearby atomic orbitals[13]; our observation of a broad range of electride character further supports that intermediate electride characteristics are possible and, more generally, electride character is tunable.

Furthermore, we propose that the structure types of these materials can be used to identify additional electride candidates, much like how Ho_2C corresponds to a wide array of isostructural electrides.[2] For example, BaAlGeF (AgLaOS-type) has 18 isostructural materials that are within the dataset – but all were excluded due to our strict criterion for volume change (< ±10%). Further investigation of each structure type can lead to more promising candidates for defluorination that may have been excluded otherwise.

Lastly, we note that all of our candidate materials are predicted to be metastable (9 – 92 meV above the hull). Even the experimentally-synthesized phase for Ho_2C is predicted to be +25 meV, which highlights the synthetic plausibility of these compounds and the margin of error for our calculations. Nevertheless, these findings are consistent with our expectation that defluorination (and, more generally, dehalogenation) leads primarily to chemically strained geometries that are conducive to electride formation.

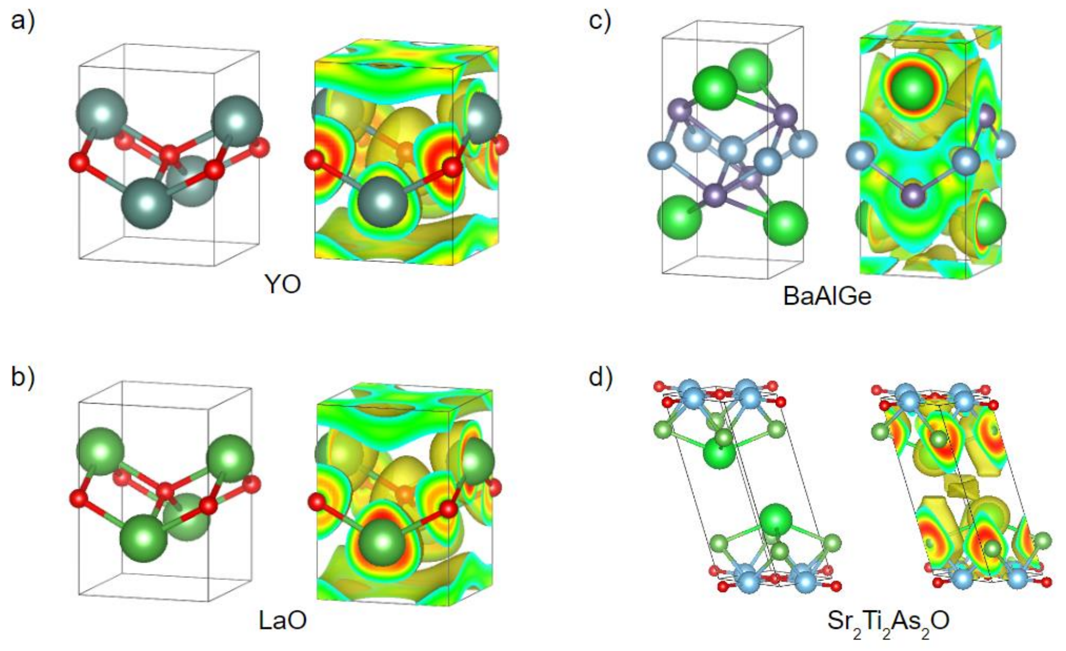


Figure 5-4. Candidate electride materials to target via low-temperature defluorination. These four materials were isolated from the initial pool of 233 structures that each began with their fluorinated analog. For each crystal structure (a-d), the left image is the bare crystal showing the layered structure, while the right image is the same crystal with an ELF isosurface applied. In each crystal's interlayer space, localized electron density is observed, which is expected for electrides.

5.7 Conclusions

Overall, we have outlined a novel synthetic strategy for the realization of electrides through low-temperature dehalogenation and illustrated the high-throughput prediction of candidate materials for this approach. By focusing on a previous report of fluoride-ion conductors[18], we analyzed >200 candidate fluorides for their potential to form metastable electrides. This resulted in several novel systems and structure types to be tested via our low-temperature synthetic approach. Furthermore, our findings indicate that the results are largely limited by the scope of known fluorides, and theoretical prediction of similar systems can also lead to promising candidate materials. Our synthetic and search strategies are generalizable beyond fluoride systems, and we now have strong evidence that many more metastable electrides exist within other halides and hydrides.

REFERENCES

(1) Liu, C., Nikolaev, S.A., Ren, W., & Burton, L.A. Electrides: a review. *J. Mater. Chem. C* **8**, 10551-10567 (2020).

(2) Druffel, D.L. et al. Electrons on the surface of 2D materials: from layered electrides to 2D electrenes. *J. Mater. Chem. C* **5**, 43, 11196-11213 (2017).

(3) Hosono, H. & Kitano, M. Advances in Materials and Applications of Inorganic Electrides. *Chem. Rev.* **121**, 5, 3121–3185 (2021).

(4) Lee, K., Kim, S. W., Toda, Y., Matsuishi, S., & Hosono, H. Dicalcium nitride as a two-dimensional electride with an anionic electron layer. *Nature* **494**, 7437, 336–340 (2013).

(5) Kitano, M. et al. Essential role of hydride ion in ruthenium-based ammonia synthesis catalysts. *Chem. Sci.* **7**, 7, 4036-4043 (2016).

(6) Kuganathan, N.; Hosono, H.; Shluger, A. L.; Sushko, P. V. Enhanced N_2 Dissociation on Ru-Loaded Inorganic Electride. *J. Am. Chem. Soc.* **136**, 2216– 2219 (2014).

(7) Kim, S. et al. Long-Range Lattice Engineering of $MoTe_2$ by a 2D Electride. *Nano letters* **17**, 6, 3363–3368 (2017).

(8) Druffel, D. L. et al. First-principles prediction of electrochemical electron-anion exchange: ion insertion without redox. *J. Phys. Chem. Lett.* **11**, 9210–9214 (2020).

(9) Burton, L.A. et al. High-Throughput Identification of Electrides from All Known Inorganic Materials. *Chem. Mater.* **30**, 21, 7521–7526 (2018).

(10) Zhu, Q., Frolov, T., & Choudhary, K. Computational Discovery of Inorganic Electrides from an Automated Screening. *Matter* **1**, 5, 1293-1303 (2019).

(11) Zhang, Y. et al. Computer-Assisted Inverse Design of Inorganic Electrides. *Phys. Rev. X* **7**, 011017 (2017).

(12) Inoshita, T. et al. Exploration for Two-Dimensional Electrides via Database Screening and Ab Initio Calculation. *Phys. Rev. X* **4**, 031023 (2014).

(13) McRae, L.M. et al. Sc_2C, a 2D Semiconducting Electride. *J. Am. Chem. Soc.* **144**, 24, 10862-10869 (2022).

(14) Zang, X. et al. Two-Dimensional Transition-Metal Electride Y_2C. *Chem. Mater.* **26**, 22, 6638–6643 (2014).

(15) Yang, L. et al. The unconventionally stoichiometric compounds in the Na–K system at high pressures. *Comp. Mat. Sci.* **200**, 110818, 0927-0256 (2021).

(16) Miao, M.S. & Hoffman, R. High Pressure Electrides: A Predictive Chemical and Physical Theory. *Acc. Chem. Res.* **47**, 4, 1311–1317 (2014).

(17) Mio, M.S. & Hoffman, R. High-Pressure Electrides: The Chemical Nature of Interstitial Quasiatoms. *J. Am. Chem. Soc.* **137**, 10, 3631–3637 (2015).

(18) Sundberg, J.D., Druffel, D.L., McRae, L.M. et al. High-throughput discovery of fluoride-ion conductors via a decoupled, dynamic, and iterative (DDI) framework. *npj Comput Mater* **8**, 106 (2022).

(19) Sun, W. et al. The thermodynamic scale of inorganic crystalline metastability. *Sci. Adv.* **2**, 11, e1600225 (2016).

(20) Fedorov, P.P. and Alexandrov, A.A. Synthesis of inorganic fluorides in molten salt fluxes and ionic liquid mediums. *J Fluorine Chem* **227**, 109374, 0022-1139 (2019).

(21) Druffel, D.L., Lanetti, M.G., Sundberg, J.D, et al. Synthesis and Electronic Structure of a 3D Crystalline Stack of MXene-Like Sheets. *Chem. Mater.* **31**, 23, 9788–9796 (2019).

(22) Sheppard, D., Terrell, R. & Henkelman, G. Optimization methods for finding minimum energy paths. *J. Chem. Phys.* **128**, 134106 (2008).

(23) Rong, Z., Kitchaev, D., Canepa, P., Huang, W. & Ceder, G. An efficient algorithm for finding the minimum energy path for cation migration in ionic materials. *J. Chem. Phys.* **145**, 074112 (2016).

(24) Larsen, P. M., Pandey, M., Strange, M. & Jacobsen, K. W. Definition of a scoring parameter to identify low-dimensional materials components. *Phys. Rev. Materials* **3**, 034003 (2019).

(25) Jain, A. et al. Commentary: The Materials Project: A materials genome approach to accelerating materials innovation. *APL Mater.* **1**, 011002 (2013).

(26) Sundberg, J.D; Benjamin, S.S.; McRae, L.M.; & Warren, S.C. Simmate: a framework for materials science. *JOSS* **7**, 75, 4364 (2022).

(27) Obrovac, M. N. and Chevrier, V. L. Alloy Negative Electrodes for Li-Ion Batteries. *Chem. Rev.* **114**, 23, 11444–11502 (2014).

(28) Ko, M., Chae, S., and Cho, J. Challenges in Accommodating Volume Change of Si Anodes for Li-Ion Batteries. *Chem Electro Chem* **2,** 11, 1645–1651 (2015).

(29) Rakshit, S., et al. Measurement of Volume Changes and Associated Stresses in Ge Electrodes Due to Na/Na^+ Redox Reactions. *J. Electrochem. Soc.* **168,** 010504 (2021).

(30) Ong, S. P. et al. Python Materials Genomics (pymatgen): A robust, open-source python library for materials analysis. *Comp. Mater. Sci.* **68**, 314–319 (2013).

(31) Pan, H., Ganose, A. M., Horton, et al. Benchmarking coordination number prediction algorithms on inorganic crystal structures. *Inorg Chem* **60**, 3, 1590-1603 (2021).

(32) Ward, L. et al. Matminer: An open source toolkit for materials data mining. *Comp. Mater. Sci.* **152**, 60–69 (2018).

(33) Atoji, M. Neutron-diffraction study of Ho_2C at 4–296 K. *J. Chem. Phys.* 74, 1893 (1981).

(34) Walsh, A., Sokol, A.A., Buckeridge, J. et al. Oxidation states and ionicity. *Nature Mater* **17**, 958–964 (2018).

(35) Dale, S.G. and Johnson, E.R. Theoretical Descriptors of Electrides. *J. Phys. Chem. A* **122**, 49, 9371–9391 (2018).

(36) Silvi, B., Savin, A. Classification of chemical bonds based on topological analysis of electron localization functions. *Nature* **371**, 683–686 (1994).

(37) Henkelman G., Arnaldsson A., & Jónsson H. A fast and robust algorithm for Bader decomposition of charge density, *Comput. Mater. Sci.* **36**, 354-360 (2006).

CHAPTER 6: ASYNCRONOUS EVOLUTIONARY STRUCTURE PREDICTION DRIVEN BY INFORMATICS AND THIRD-PARTY DATABASE INTEGRATIONS

6.1 Summary

Evolutionary search algorithms have become an essential tool for materials discovery because of their ability to predict thermodynamically favored phases using only composition as an input. Unfortunately, evolutionary structure prediction involves thousands of local DFT optimizations that begin from fully random inputs, and existing implementations struggle to share results across separate runs. Here, we introduce a new software for evolutionary search strategy integrated with the Simmate framework. Simmate handles data management and workflow orchestration and allows us to introduce (a) improved starting points using third-party crystal databases, (b) data sharing across separate evolutionary searches, and (c) asynchronous scaling and distribution of calculations across arbitrary resources. Even without seed structures and cross-search data sharing, our new search algorithm led to a speedup of over 10x for structure prediction of fixed compositions relative to leading evolutionary codes. Furthermore, we estimate another order of magnitude improvement when seeds and cross-search features are active. Our search can scale to complex chemical systems and to over 11 million structures (+ energies and site forces) per week. Our evolutionary search framework will greatly accelerate phase space exploration, facilitate massive collaborative efforts/result sharing, and lead to the largest crystal dataset yet developed. We anticipate that this dataset will be leveraged for many far-reaching applications.

6.2 Introduction

Despite the many years of material exploration, the vast majority of unique crystalline materials – likely over 99.9% – remain undiscovered.[1,2] These undiscovered materials may offer stronger steels, better catalysts, improved transistors, and many other solutions to urgent societal problems.[3] We therefore need a fast and efficient way of identifying new materials so that society can harness their benefits.

Currently, the prediction of novel material structures most widely employs evolutionary search algorithms, which require thousands of ab initio calculations and millions of CPU hours, even for a single binary phase diagram.[4] Existing evolutionary codes struggle to explore beyond simple systems due to computational cost, and further, these codes are unable to share information across evolutionary searches without repeating calculations.[5,6,7,8,9,10] Since the release of these evolutionary codes, the larger academic community has made substantial progress on workflow orchestration and data management and has produced many large crystal databases.[11,12,13,14,15,16] However, these advances have largely remained independent from existing evolutionary materials discovery software.

Here, we introduce a new evolutionary algorithm within the Simmate framework[17], which integrates well-established packages for data management, workflow orchestration, and material science analysis. Simmate enables many novel features in the context of evolutionary structure prediction, including (a) automatic integration of third-party data and software, (b) data sharing across separate evolutionary searches, and (c) asynchronous scaling and distribution of calculations across arbitrary resources. Moreover, the automatically built database API allows us to optimize a search's core components, e.g., structure creation and transformation. Together, Simmate’s high-level framework and our new benchmarks have allowed us to accelerate the prediction of binary systems by over an order of magnitude relative to other search algorithms.

Within this chapter, we first provide an overview of the structure prediction challenge, existing prediction strategies, and the core components of an evolutionary search. Using this background information as our reference point, we then describe our search strategy using the ground-state prediction for individual compositions, followed by more intensive “variable-composition” calculations that explore a full chemical system. We then report our benchmarks and comparisons of core components across all evolutionary search algorithms and describe how they influence search speed/efficiency. Finally, we provide a perspective on how our new search algorithm and resulting database can accelerate complementary fields of materials science.

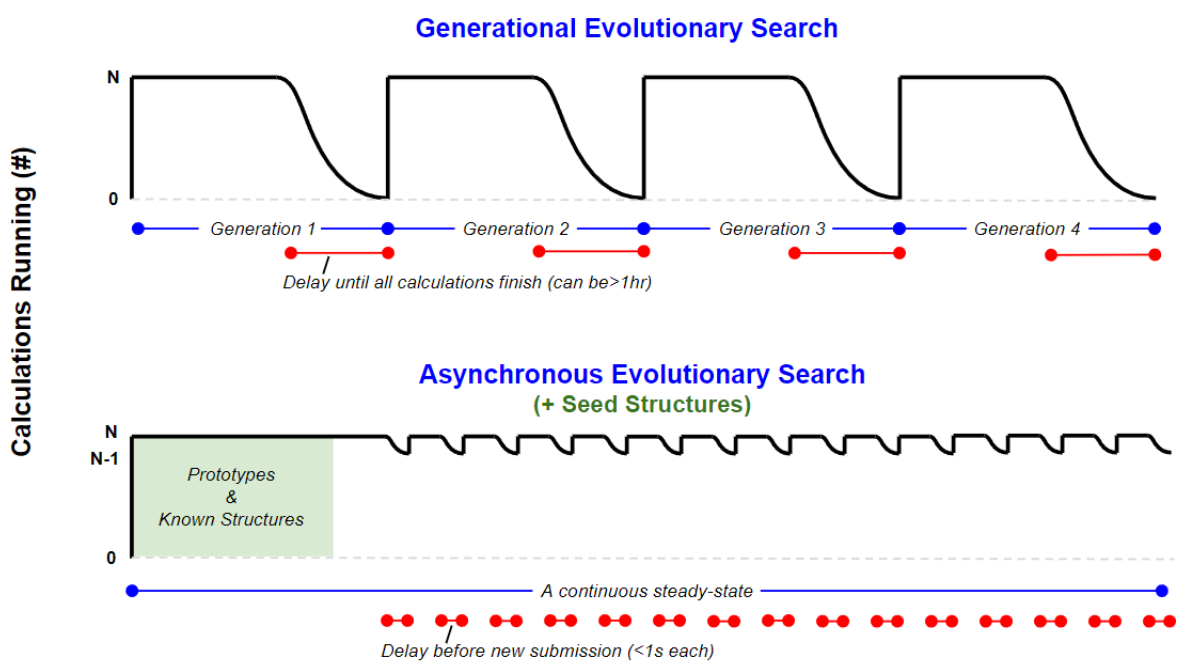

Generational Evolutionary Search
Calculations Running (#)
N
0
Generation 1
Generation 2
Generation 3
Generation 4
Delay until all calculations finish (can be>1hr)
Asynchronous Evolutionary Search
(+ Seed Structures)
N
N-1
0
Prototypes
&
Known Structures
A continuous steady-state
Delay before new submission (<1s each)

Figure 6-1. Illustration of generational and asynchronous evolutionary searches. Generational searches (top) batch submit N structures, wait for all calculations to finish before the results are analyzed, and then submit a new batch of structures (i.e., a new generation). The waiting step causes computational resources to hang until the slowest calculation finishes, leading to inefficient use of resources. This work introduces an asynchronous search with seed structures (bottom). Rather than beginning a search with random guesses, seed structures are created from known materials and prototypes pulled from popular crystal databases. These improve the search's starting point. Whenever a calculation is completed, it is immediately replaced with a new submission to maintain a constant number of calculations in the queue. Importantly, to maximize information transfer, structure creation (and parent selection) is performed at run time rather than before job submission.

6.3 An overview of phase space complexity

Most materials remain undiscovered due to the scale of compositional space that needs to be searched along with the slow rate at which researchers are capable of characterizing a single composition. To illustrate this issue, we can look at a ternary system such as the scandium-carbide-fluoride system (Sc-C-F). Here, even when we exclude disordered phases and only consider highly crystalline phases (< 30 atoms in the unit cell structure), we are still left with 4,306 unique reduced compositions ($Sc_xC_yF_z$) to consider. Experimental exploration of a single composition (e.g., Sc_2C) can take many years. This is exemplified by the identification of the competing phases and thermodynamic stability of Sc_2C, which required many studies carried out across several different decades.[18]

With everything above considered, we can quickly realize that detailed exploration of all compositions in a single ternary system is not feasible. Exploration must therefore be guided by chemical intuition and computational predictions. The necessity of guided exploration is exacerbated when we consider the number of possible ternary systems (~1 million systems for 100 unique elements) and how compositional space complexity scales exponentially for systems with more than three elements (e.g., ~9 billion unique systems with 5 different elements).[19] Experimental exploration is too slow here, so a rapid and systematic way of computationally predicting desirable phases is essential.

Moreover, the computational exploration of even a single composition represents an enormous challenge. This is due to the high number of degrees of freedom available when creating potential structures: 6 from lattice vector lengths/angles and 3×N for each xyz coordinate of N total atoms. Even when the problem is simplified, there are 10^{11} (100 billion) possible configurations for a fixed lattice (10 Å^3/atom), discrete atomic sites separated by 1 Å, and 10 atoms of a single element (A_{10}).[19] This scales exponentially to 10^{47} configurations for a two-element $A_{15}B_{15}$ composition. Consideration of all configurations is simply not possible.

The number of possible compositions – combined with the complexity of a single one – illustrates the core challenge of materials discovery and exploration: how can we efficiently explore and identify systems? The next generation of high-performance materials may come from yet undiscovered compounds, so we must begin to address this challenge.

6.4 Existing search strategies

As a result of phase space complexity, the computational prediction of a composition's thermodynamically favored phase has been a topic of extreme interest. There are many diverse strategies – ranging from prototype libraries[20,21,22] to Monte Carlo sampling[5] – that 'search' for the lowest energy structure. These existing search strategies can be evaluated on (i) whether a search starts in a promising region of phase space, (ii) to what degree a search can improve on the best structure as it progresses, and (iii) the overall computational cost of the search.

The most straightforward structure prediction strategies involve the creation of best-guess structures from existing materials and prototypes. It is well-established that similar compositions often possess equivalent phases, e.g., many +1 cation / -1 anion solids possess a rocksalt structure type. It is therefore very effective to take known systems and substitute them into a desired composition. This has given rise to large initiatives such as the OQMD and AFLOW databases[13,14], which iterate through a library of prototype structures[20,21,22] to produce several million theoretical structures. While these strategies are computationally inexpensive and produce reasonable guesses, they are unfortunately incapable of identifying novel structure types and have no way to determine if the lowest-energy structure has been identified. They must be paired with other search strategies or experiments to add confidence to the results.

Best-guess and prototype structures can also be used for dynamics-based approaches, where a series of simulations/distortions are applied to arrive at a local thermodynamic minimum. This strategy is analogous to the experimental high-temperature annealing of a material. The popularity of these search strategies has led to many variant algorithms, including meta-dynamics[23,24], simulated annealing[25], and basin hopping[26]. Collectively, however, dynamics-based approaches cannot guarantee that the starting structure is reasonable and will only be able to improve to a nearby local minimum. In other words, there is no guarantee that the final structure is also the energetically global minimum. Furthermore, dynamics runs require large supercells, which are computationally expensive to evaluate. Dynamics-based search strategies have therefore become limited to systems where there is a high confidence in the input structure and a detailed understanding of phase transformations is desired.

In a vastly different approach, one may want to search the phase space without a biased ‘best-guess’ input. One can search phase space through Monte Carlo sampling of randomly created structures, paired with a mechanism to select and improve upon best structures. For example, evolutionary structure prediction iteratively mutates best structures until the ground state phase is found.[19] Particle swarm optimization and other genetic algorithms behave in a similar manner, where (generally) the method of structure selection and mutation subtly varies between each approach.[27] This group of search algorithms is very effective at evaluating the promising regions of phase space on the fly and, more importantly, improving the best candidate structures in multiple regions of phase space. As a result, they perform the best in blind tests for structure prediction when compared to full Monte Carlo and dynamics-based strategies.[28] However, despite their widespread use and success in materials discovery, genetic algorithms still require the evaluation of hundreds of crystal structures for a single composition and thousands of structures when evaluating full chemical systems. The robustness of this technique therefore comes at a high computational cost.

In summary, there are many approaches to predict and search for ground state structures – each with differing total computational costs and confidence of results. The material science community has most recently begun combining the advantages of each of these different methods, but there is still no all-in-one solution to complex phase space exploration.

6.5 Evolutionary Structure Prediction

Evolutionary structure prediction applies the concept of biological natural selection when exploring materials phase space: (a) initial structures are randomly generated and evaluated for their thermodynamic stability (as a measure of "survival fitness"), (b) the best of these structures are used to generate new derivative and transformed structures (emulating reproduction and potentially desirable mutations), and (c) structure creation and transformation is repeated for many "generations" until the search converges to the ideal ground state phase (Figure 1-3).

The search algorithm is straightforward from a bird's-eye view, but there are a plethora of strategies to modify the underlying components of a search – namely, how new and "parent" structures are created, evaluated, and mutated. Moreover, additional scrutiny can be placed on mechanisms for duplicate removal, convergence criteria, fitness measurement, or even data management. As a result, many evolutionary searches and genetic algorithms have been reported over the past decade. These include USPEX[6], XtalOpt[7], CALYPSO[10], GASP[9], ASE-GA[8], and AIRSS[5]. There are many similarities between each program from a high-level perspective, but their underlying components vary greatly. These components are essential in the optimization of the overall search's efficiency, which is why each program has seen drastically different levels of success and adoption by the materials community. We provide a detailed comparison of core components in Section 6.8 of this manuscript, but for the time being, we will focus on the end result of underlying component optimization: the overall search strategy and efficiency.

When looking at the overall search strategy, it is important to consider the bounds of a search. One can explore a single composition (e.g., SiO_2 or $Mg_4Si_4O_{12}$) or full chemical system (e.g., Si-O or Mg-Si-O), and in addition, the target structures can be limited to crystalline bulk materials, films, nanoparticles, or any other phase constraints. Because they are the most straightforward and best dictate the search efficiency of multi-composition systems, we begin with the prediction of crystalline bulk materials of a single composition.

6.6 Single composition search

A successful search algorithm must be able to efficiently and robustly predict the thermodynamically favored phase of a given composition and stoichiometric factor. Here, we define a composition as a specific stoichiometry (or ratio) of elements, while the total number of atoms in the final structure dictates the stoichiometric factor. For example, we could evaluate the composition of $MgSiO_3$ at different stoichiometric factors: $Mg_2Si_2O_6$ (factor=2) and $Mg_4Si_4O_{12}$ (factor=4). To evaluate our search algorithm, we use a series of nine fixed compositions that were selected from benchmarks of past reports on evolutionary search algorithms[29]: Fe_1, Si_2, C_4, Ti_2O_4, Si_4O_8, Al_4O_6, $Si_4N_4O_2$, $Sr_4Si_4N_8$, and $Mg_4Si_4O_{12}$.

6.6.1 Input and configuration

For each compositional search, our software requires minimal user configuration but enables the flexibility to scale and modify search behavior. Thanks to integration with the Simmate framework[17], a search can be submitted as a single workflow in various formats (YAML, TOML, Python, Bash, etc.). For example, we show a minimal input file in the YAML format in Figure 3-8a. The use of custom settings and methods for structure creation, transformation, and more are supported through the full list of optional parameters. Throughout this section, we discuss the default settings used, and a more in-depth discussion on advanced setting modification is available in our online documentation.

6.6.2 Starting structures

In other evolutionary search programs, a single composition search is initialized with randomly created structures of the target stoichiometric factor along with user-provided “seed” structures. Together, randomly created structures and seed structures make up the “first generation” of the candidate phase, from which the cycle of “natural selection” and “genetic mutations” will follow. Our search algorithm follows this pairing of random and seed structures, although there are important differences in how we generate these starting structures.

The first key advancement is the automatic generation of seed structures. Simmate is integrated with several third-party databases and facilitates the addition (and creation) of new datasets – meaning searches can easily begin with (i) all known theoretical or experimental phases, (ii) reasonable substitutions of known structures, and (iii) creation of structures from prototypes. This creates a batch of seed structures to submit at the beginning of an evolutionary search. Because querying external databases is only performed once and not repeated thereafter, we refer to these as “single-shot” structure sources.

For (i), we currently include structures from the COD[16], Materials Project[11], OQMD[13], and JARVIS[12] databases. We also provide guidance on how others might add new datasets from an Excel file, list of CIFs, or hosted API. We note that, in some cases, input datasets may already include the ground-state phase and experimentally known phases. Nevertheless, by still running the evolutionary search, one can add confidence to the ground state structure as well as discover new and competing phases.

For (ii), we use known structures of chemically similar compositions as templates for new inputs. For example, a search on Y_2CF_2 would predict that fluoride is chemically similar to other halogens (Br, Cl, I), and our software will query other databases for all Y_2CX_2 (X=Br, Cl, I) structures, convert the resulting structures to Y_2CF_2 via substitution, and then submit them to the search. The same can be done with carbon, yttrium, or even multiple elements at once. The matrix of potential substitutions is generated using pymatgen’s substitution module[30,31], which is based on data mining of the Materials Project database[11]. We extended the predicted compositions to structures in COD, OQMD, and JARVIS as well – increasing the number of possible matches.

For (iii), the AFLOW prototype libraries[20,21,22] are used as template structures. While several methods for structure generation are supported, the default behavior is to only grab prototypes exactly matching the anonymous formula (e.g., all AB_2C_2 prototypes for Y_2CF_2). Structures are then substituted into the target composition and submitted to the search. We note that the prototypes are limited to those available through the PyMatGen package, but there are opportunities to extend the prototype library (see Section 6.9.2).

There are also differences in our strategy used for the randomly generated input structures. While several algorithms for generating random structures exist and are supported (see section 6.8.1), our default method follows the algorithm used by PyXtal and RandSpg software packages.[7,32] Here, high symmetry structures are generated by randomly selecting possible Wyckoff site combinations from a randomly selected space group. However, the validation of the created structures differs within our algorithm. Lattice volumes and minimal atomic distances are determined from the most likely oxidation states of the composition, where initial values are based on hard sphere packing of atoms. Tolerance values can also be updated as the search progresses. Furthermore, structure validation is performed using Simmate's extension to common MatMiner featurizers[33], such as the partial radial distribution fingerprint. This ensures that a diverse pool of structures is submitted at the start of a search and duplicate structures are not submitted thereafter.

Together, automatically generated seed structures and high-symmetry random structures establish a much-improved starting point for evolutionary structure prediction. In fact, because seed structures frequently capture the ground state structure, we choose to exclude these seeds from our initial benchmarks. This helps to ensure that our evolutionary search is indeed efficient and robust when starting from ab initio inputs.

6.6.3 Workflow orchestration & database storage

Initial candidate structures are often very far from the ground state phase, so one must efficiently relax each to its local minimum. Local optimization allows individual structures to move to regions of improved phase space – greatly reducing the phase space complexity problem to regions bound by chemical constraints. For this reason, all existing evolutionary structure prediction software programs integrate with third-party programs to evaluate and locally optimize structures. This includes empirical models or ab initio calculations using programs such as LAMMPS[34], VASP[35], Quantum Espresso[36], or the many others available[37,38].

Local optimizations in evolutionary searches also introduce many unique challenges due to their scale and diverse range of inputs. Input structures are error prone due to potentially high energy and unreasonable starting points; hundreds or thousands of calculations need to be efficiently run in parallel; and the results from the plethora of data and output files (sometimes >1 TB across thousands of relaxations) must be efficiently and effectively stored. Thanks to Simmate's workflow and database framework, these challenges are much easier to address.

Simmate's workflow framework allows integration of any user-defined workflow and is particularly well-equipped for calling external file-based programs with monitoring and error recovery. At the time of writing, Simmate primarily includes scientific workflows that use the VASP software[35], where the majority of workflows are direct equivalents to those used by the Materials Project[39]. Furthermore, full reimplementation of the Materials Project's error handlers for automatic error detection and recovery is also included.[40] By default, each structure in an evolutionary search undergoes a series of five relaxations that gradually increase in quality and whose parameters are dynamically determined based on input structure and composition. Together, this workflow setup means that minimal user configuration is required, relaxations are robust to common errors, new workflows can be easily incorporated, and advanced setup/monitoring features can be quickly appended to workflows.

In addition to robust individual structure runs, the challenges of scale and orchestration are also effectively addressed with the Simmate framework. Typically, evolutionary software will have users specify and configure a high-performance computing (HPC) cluster queue system (e.g., SLURM or PBS) and then submit specific structures/files to the system. This requires configuration of computational resources and a direct connection between the search resources – often limiting computational resources to those that are (i) officially supported by the team of the evolutionary-search software and (ii) resources that share a file system. Instead, Simmate's workflow engine is agnostic to where and what computational resources are used. Here, resources communicate indirectly through the database backend. This means that a new resource can be started anywhere, requires only an internet connection, and does not require any prior knowledge of submitted workflows or searches. Searches can therefore distribute parallel calculations across multiple HPC clusters, user desktops, and/or cloud computing services. Resources are added by simply calling a 'start-worker' command (worker = a resource that runs scheduled workflows), and workers can be added or removed at any point during the search without affecting search behavior. Thus, Simmate allows evolutionary searches to run on a single computer and then quickly scale to massively parallel and diverse setups.

Lastly, Simmate facilitates storage and access of relaxation results. Evolutionary search programs frequently store only essential thermodynamic information and remove output files to save on disk space. However, we find that the large amount of data produced by evolutionary searches can have far-reaching applications in materials science, such as the production of a community database of diverse structures, atomic forces, and energies (see Section 6.9.2). We therefore sought to optimize result archives and build database APIs. Each workflow can optionally remove duplicate data and output files (e.g., potential files) and compress outputs to archives (e.g., zip or tar files). Furthermore, Simmate's database models automatically populate the database with common filtering criteria and build our website interfaces, REST APIs, and Python ORM. As a result, data can be simultaneously used (and added to) from multiple searches or even separate studies. In other words, regardless of what led to the submission of a workflow, its results can be used across any search or follow-up analysis. Cloud database integration therefore plays a central role in optimizing data analysis and reuse.

Together, workflow orchestration and database storage empower the many local optimizations of an evolutionary search. The ability to recover from errors, scale across arbitrary resources and share results between analyses has diverse implications – both in how a search is performed and how results enable new independent studies.

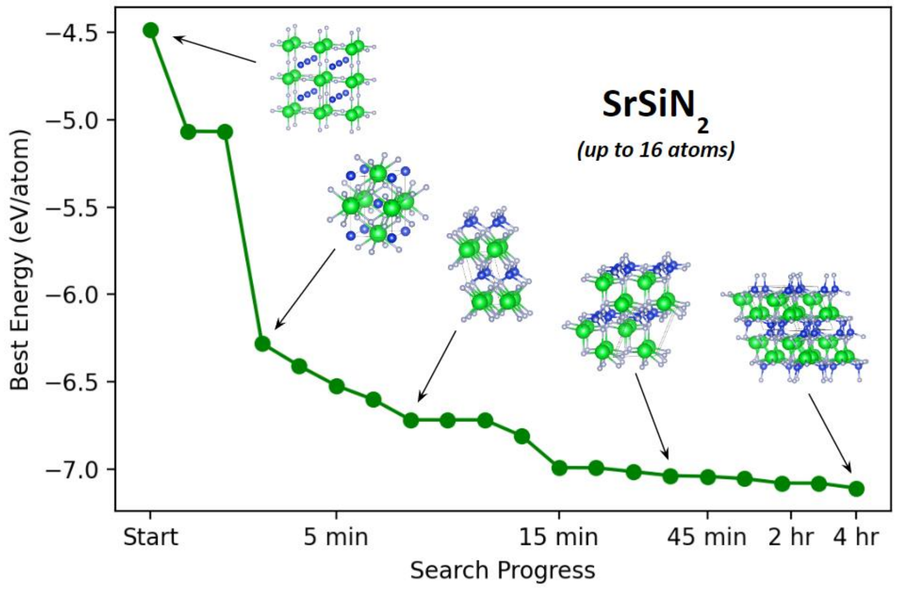


Figure 6-2. Evolutionary search convergence for $Sr_4Si_4N_8$. An evolutionary search without seed structures begins with random structure creation, which often produces high energy and unreasonable phases (left). The evolutionary cycle of creation and transformations gradually improves on the best structure(s) and ultimately leads to the thermodynamic ground state structure (right).

6.6.4 Progression of new structures

Up to this point, we have established the initial inputs and a working framework to locally optimize structures and manage results. We now turn to how our search progresses and arrives at improved structures. The primary mechanism of evolutionary search algorithms is the creation of new structures from the 'best fitness' individuals who have been previously evaluated – and repetition of this process to convergence (Figure 6-2). Thus, we need to continuously take the best structures and attempt to improve them through transformations.

The iterative cycle is typically done in batches (i.e., a "generation" of structures), but this can lead to underutilization of computation resources. Specifically, evolutionary algorithms will submit a group of structures and then wait until all calculations complete before submitting the next group of improved structures. Separate from the materials science community, asynchronous evolutionary searches have been shown to increase candidate throughput by up to 75% and to have searches converge an order of magnitude faster.[41] Here, a fixed number of scheduled workflows is maintained in the job queue at all times, which we refer to as a "steady state" of candidate structures. Whenever a workflow is completed, a new workflow is immediately scheduled to replace it. At no point does the search block wait for a group of structures to finish and, as a result, the downtime of computational resources is minimized. We therefore adopt this asynchronous search strategy.

In an asynchronous search strategy, it is important to consider when parent structures are being selected and when new transformed structures are being generated. Structure generation prior to job submission is suboptimal because it does not leverage new data acquired while the workflow was in the job queue. To illustrate, let us say we generated and submitted a structure that sits in the job queue for >1 hour. During this queue time, other calculations are running that may contribute to the evolutionary search and may produce new best structures, and in hindsight, we would have preferred to generate our structure using these new best structures. Therefore, our search generates new structures only at the instant before they are needed so that the utilize the most current data available. Our steady state search of structures therefore follows a dynamic, decoupled, and iterative (DDI) search strategy, which we have reported elsewhere.[42]

The total number of steady-state structures is maintained through a combination of structure creation and transformation sources. In contrast to "single-shot" sources that are submitted once and then never again, steady-state sources are submitted continuously until the search converges. For example, one simple transformation that can be applied to best structures is a coordinate perturbation (i.e., small atomic displacements to break cell symmetry) and, by default, our search will maintain five submitted calculations of this type in the queue at all times. As soon as one structure finishes, a new structure is submitted immediately to take its place.

Other transformations include heredity, soft mutation, mirror mutation, lattice strain, rotational mutation, atomic permutation, and extreme symmetry reduction. The majority of these transformations are built off functionality in the ASE[43] and PyMatGen[30] packages and are commonly seen across all evolutionary structure prediction codes. A steady state of randomly created structures is also maintained throughout the search, as this prevents the search from becoming trapped in a local minimum. Together, random creation and the pool of transformations are used to generate new structures that progress the search.

We note, however, that there are additional steady-state sources active when the dataset contains results from other searches. For example, search results from chemically similar compositions can be used to accelerate another ongoing search. The most straightforward of these is identical to our substitution-based inputs, where template structures are pulled from other evolutionary search results, rather than third-party databases. When active, this means that a key finding from one search will propagate and accelerate other active searches. Other, more complex cross-search transformations are possible with variable composition search, which we discuss later. Even though these are among the most successful and important transformations in our search algorithm, we exclude these structure sources during our single composition benchmarks. This is because they are not immediately active in a clean Simmate installation, and further, we wish to benchmark from ab initio inputs. When no data are available, these sources silently shut down and do not affect the search algorithm.

In summary, there is a diverse combination of random creations, transformations, and cross-search collaboration occurring throughout a single-composition search. While it is possible to adjust the steady-state amount of each source during the search, dynamic updating of these settings is not yet automated, as some evolutionary codes support.[44] We find that the default values effectively converge searches toward the ground state structure and report on the search efficiency of this setup herein.

6.6.5 Search results

We evaluate the efficiency and effectiveness of our search algorithm using total CPU time and real time, rather than past reports on evolutionary search algorithms that emphasize the total number of structures evaluated. These times better represent the practical usefulness of the overall method and how rapidly the search algorithm is able to progress. Furthermore, we have found that search speed can be improved significantly with the structure generation and optimization methods, where the total number of structures metric does not capture these improvements. Nevertheless, we still report the total number of structures required for our single composition searches (Fe_1, Si_2, C_4, Ti_2O_4, Si_4O_8, Al_4O_6, $Si_4N_4O_2$, $Sr_4Si_4N_8$, and $Mg_4Si_4O_{12}$), but we emphasize that fixed-composition searches are slower and less desirable than the variable composition searches discussed in the following section. Single composition searches give us a starting point to evaluate how long our evolutionary search algorithm takes.

The end of a search – specifically, the stopping of steady-state sources – is determined by several possible stop conditions. The primary stopping criterion checks for convergence of structure energies, where the search is considered converged if the best structure does not improve by a set tolerance (e.g., 1 meV/atom) after the evaluation of many new structures (e.g., 200 new structures). However, to rapidly benchmark our search, the user may provide an expected structure as an input, and the search is stopped when that structure appears in the search results. Herein, we report evolutionary search results on our benchmark compositions up to the 'expected structure' stop condition.

A series of fixed composition searches are carried out to determine the search speed and efficiency. Each composition is evaluated 10 times using an independent and empty database, and the search results are reported in Table 6-1. We also compare these search results to equivalent searches using USPEX (v10.5.0) (Figure 6-3). Our results indicate that Simmate is faster than USPEX by factors of 8x to 39x for smaller systems (<20 atoms). Further, it appears that USPEX is prone to phase trapping in >10 atoms, causing searches to fail at finding the ground state phase. For example, Simmate found the $Si_4N_4O_2$ ground state phase in ~20 minutes on average, while USPEX failed to find the ground state after ~3 days.

Even with this massive speedup from Simmate, we emphasize that there is plenty of room to improve. We found that in larger systems, Simmate would rapidly identify promising regions of phase space, but improvements within promising regions was significantly slower. This is observed in a search for $SrSiN_2$ shown in Figure 6-2, where final stages of the search correspond to >50% of the calculation time. This indicates that Simmate requires more intelligent transformations or simply optimization of existing ones. Thus, we find that Simmate is better optimized for phase space exploration of crystalline bulk phases (<20 atoms), and furthermore, Simmate requires optimization of structure transformations before high atom counts (>30 atoms) are recommended.

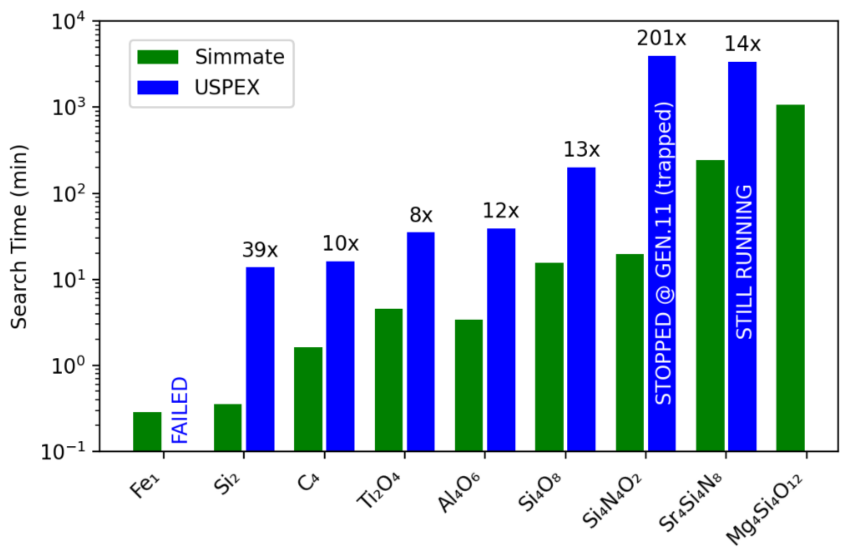


Figure 6-3. Comparison of search times for Simmate (v0.12.0) and USPEX (v10.5.0). All searches are parallelized across 100 jobs, where N=100 corresponds to the steady state in Simmate and the generation size in USPEX. The relative search time of T_{USPEX} vs. $T_{Simmate}$ is given above the USPEX entries. For the USPEX searches: Fe_1 failed due to structure creation; $Sr_4Si_4O_2$ was terminated after the 11th generation because the search was trapped without ever finding the ground state; and the $Sr_4Si_4N_8$ search is still actively running at the time of submission but has not yet found the ground state.

Table 6-1. Benchmark evolutionary searches for fixed compositions without seeds or cross-search transformations. All searches were parallelized over 100 jobs with 4 cores each, and the reported times reflect when the target structure was first identified. 10 searches were performed for each composition, and metrics below are the mean values.

Composition	Target structure (MP-id)	Atoms per structure (#)	Structures until ground state (#)	Total real time (min)	Total CPU time (hr)
Fe_1	mp-13	1	6	0.28	0.09
Si_2	mp-139	2	19	0.35	0.35
C_4	mp-48	4	116	1.61	4.24
Ti_2O_4	mp-390	6	146	4.56	14.50
Al_4O_6	mp-1143	10	74	3.39	8.01
Si_4O_8	mp-6945	12	312	15.61	64.68
$Si_4N_4O_2$	mp-4497	10	388	19.74	89.82
$Sr_4Si_4N_8$	mp-4549	16	1118	242.28	1355.57
$Mg_4Si_4O_{12}$	mp-603930	20	5755	1073.57	6212.00

6.7 Variable composition searches.

Predicting the ground state phase of a specific composition and stoichiometric factor is only one part of the materials discovery challenge, as we must still address the exponential growth of phase space complexity and the full range of possible stoichiometries in variable compositions. In the simplest case, we must consider a single composition search done with a variable number of sites in each structure (e.g., $Mg_xSi_xO_{3x}$ where x=1, 2, 3, …). However, to identify exciting and unexpected phases, we must also efficiently explore full chemical systems (e.g., $Mg_xSi_yO_z$, where x, y, and z each range from 0-1). Running a full fixed composition on the millions of possible compositions would not be possible. We must therefore reevaluate our search strategy for complex variable compositions.

We find that phase space exploration can be addressed by considering two factors: (1) both phase space complexity and computational cost scale exponentially with the number of atoms in the structure, and (2) similar compositions will possess similar phases and search results. These points suggest that (a) we can learn about a system more rapidly by evaluating many smaller structures before spending time on larger ones and that (b) we can leverage results from computationally cheaper and similar searches to speed up more-complex searches. As a result, we propose that the most efficient way to explore complex phase space is to begin with simple, high-symmetry compositions and systematically transition to more complex, low-symmetry phases over time.

Our systematic strategy is starkly different from those used by existing materials prediction programs, which instead rely much more heavily on evolutionary search concepts.[4,6] We elaborate on our approach and its differences by looking at three variable composition searches of increasing complexity: a composition with variable number of sites, a full binary chemical system, and a full ternary chemical system.

6.7.1 Variable number of sites

When other evolutionary algorithms implement searches with a variable number of sites, very little is changed from the fixed composition search. The most common strategy is to introduce initial structures with a distribution of site counts and new mutations that manipulate site counts. However, this introduces expensive calculations at the beginning of the search, which slows preliminary phase exploration.

Instead, we begin our search with the smallest stoichiometric factor and systematically work up to the target factor. For example, a search on $Mg_4Si_4O_{12}$ would begin by running a fixed-composition search on $MgSiO_3$, followed by searches on $Mg_2Si_2O_6$, $Mg_3Si_3O_9$, and finally $Mg_4Si_4O_{12}$. Thus, a “variable-nsites” search (nsites = number of atomic sites) is effectively a series of fixed composition searches of increasing size. This is done because the computational cost of relaxing individual structures typically scales with the number of atoms in the structure cubed (N^3). It is therefore much more efficient to explore phase space using smaller structures in the beginning and, once promising regions of phase space are identified, adjust the search for lower-symmetry phases.

However, there is one exception to our stepwise approach: the point at which variable-nsite seed structures are submitted. Here, seed structures for all stoichiometric factors are submitted up front because they are often the best candidate materials. The variable composition search will then progress systematically after these seed structures, regardless of which ones result in the lowest energy structures. This ensures that prototypes and known structures for larger cells are calculated immediately, rather than waiting for earlier search stages to finish.

This stepwise approach is successful only if information can be passed between each stage of the search, e.g., the results of the $MgSiO_3$ search must be used to speed up the $Mg_2Si_2O_6$ search, which is in turn used for the $Mg_3Si_3O_9$ search. Currently, this is done through several steady-state transformation sources. Structures with fewer sites are either combined to generate larger cells (i.e., structure splicing) or transformed into supercell derivatives. These transformations ensure that the most promising local geometries are maintained while long-range ordering is exploredWith these new transformations, the progression of new structures builds off those described in the fixed composition search (e.g., cross-search collaboration and common transformations).

6.7.2 Binary system search

Searches for full binary systems are also carried out in a stepwise fashion (Figure 6-1). Similar to variable-nsite searches, a binary system search involves a series of fixed-composition searches that are performed with increasing site counts (N) and phase complexity. For example, the yttrium-carbon system involves running the following fixed-composition searches in order: N=1 (Y_1, C_1), N=2 (Y_2, Y_1C_1, C_2), N=3 (Y_3, Y_2C_1, Y_1C_2, C_3), N=4 (Y_4, Y_3C_1, Y_2C_2, Y_1C_3, C_4), and so on. This process continues up to a user-specified maximum number of allowed sites. Similar to the variable-nsite search, we also submit seed structures for all possible compositions before any search is started. Our approach utilizes rapid, simple searches to accelerate the exploration of more complex systems.

In addition to transformations from fixed and variable-nsite compositions, binary systems are made more efficient via cross-composition mutations. These mutations are particularly successful because similar compositions frequently have similar structures. For example, it is likely that the ground state structure for $Y_{10}C_9$ is a disordered form of the $Y_{1x}C_{1x}$ ground state structure. New transformations therefore pull from nearby compositions and carry out element permutations, random site removals, and random substitutions.

An additional strategy to improve the search efficiency is to identify specific categories of wasteful DFT calculations from conventional evolutionary algorithms and prevent them from running. This includes (i) compositions where the ground state structure is already known, (ii) low-symmetry structures, and (iii) unstable compositions. To address (i), our search algorithm skips single element compositions by default, instead making them reliant on seed structures. For (ii), time may also be wasted on compositions with high stoichiometric factors (e.g., $Y_{20}C_{20}$, factor=20) because the ground state structure is typically higher symmetry and was found in earlier searches. We therefore limit searches to a maximum stoichiometric factor (the default is factor=4) and encourage users to run more robust fixed-composition searches after the binary search completes. And for (iii), unstable compositions can be optionally skipped/stopped if we do not expect to find a stable structure. In other words, we can skip a composition if no structures are found within 500 meV/atom of the hull. Currently, however, this condition is not applied by default because reliable cutoffs are still being tested. Together, these considerations allow us to focus on the most promising and truly exploratory compositions.

Even though our smaller fixed-composition searches utilize an evolutionary algorithm, our overall strategy for the binary system search should not be considered evolutionary. Specifically, our individual fixed-composition searches are evolutionary searches, while the collective search of the full chemical system is not. This is different from existing search algorithms, which explore binary systems by evaluating all compositions simultaneously.[4,6] Instead, we opted for a systematic implementation to better control which compositions and phases are explored. We anticipate that this will lead to more efficient exploration of binary phase space.

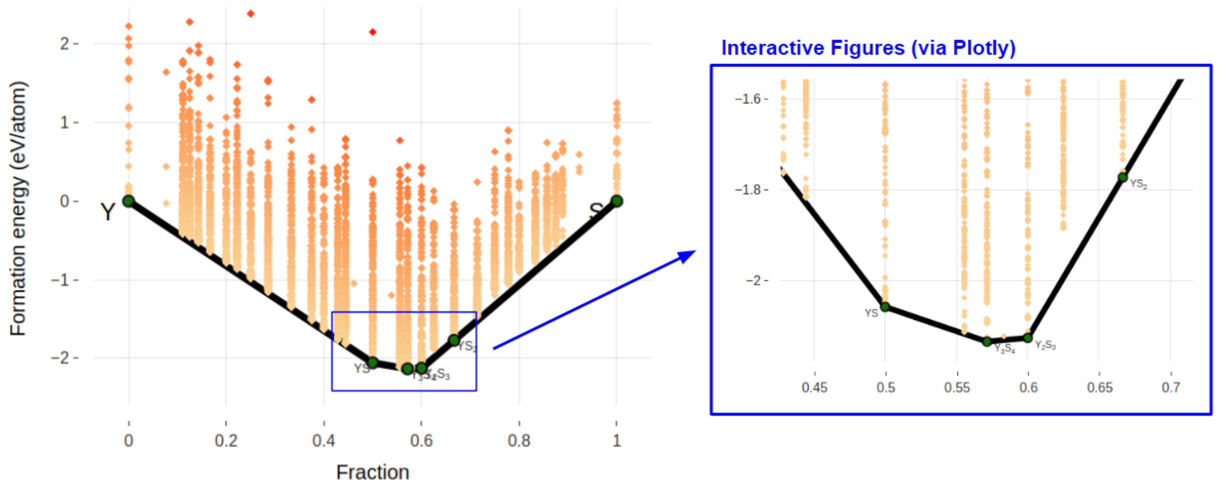


Figure 6-4. Example hull diagram from a binary-system search. Binary searches are carried out as a series of fixed-composition searches of increasing complexity, where each fixed search corresponds to a yellow/orange vertical line of sample structures on the hull diagram. The results are plotted using the PyMatGen library, which outputs figures and interactive Plotly figures.

6.7.3 Complex and >2 element chemical systems

It is also possible to explore chemical systems with more than 2 elements using our software, but we are not yet able to dynamically prioritize promising regions of phase space. For example, while exploring a ternary system such as Y-C-F, we might quickly learn that compositions such as YC_8F_x are unlikely to give stable phases, whereas compositions such as Y_2CF_x would be much more likely to be on the hull. Currently, ternary systems are explored in the same manner as our binary systems, where all compositions are evaluated in order of increasing atom count. Our software is still able to rapidly explore these systems, but because we do so in a brute-force way, we outline promising systematic search strategies that can be used to accelerate the current algorithm.

Before discussing these systems, we first want to clarify what is meant by complex chemical systems. Here, complex systems are the connection or linear combination of multiple compositions, as opposed to pure elements. For example, $(Y_2C)_x(Y_2CF_2)_{1-x}$ could be considered a complex composition. Other complex systems can also be drawn out using likely reagents. For example, Y-C-YF_3 would be a complex ternary system corresponding to a search for $Y_{x+z}C_yF_{3z}$ phases. Together, these complex systems help define compositional boundaries to explore within larger systems.

In the previous section (6.7.2), we found that a systematic search leads to significant speedups in binary system exploration, relative to full evolutionary searches. Ternary and other complex chemical systems (e.g., Y-C-F and Y_2C-Y_2CF_2, respectively) are equivalent to a series of many binary systems, so we expect an even larger speed up for >2 element searches. Indeed, it has been previously reported that systematic exploration of ternary systems is successful where full evolutionary searches are not.[44] However, the optimal systematic search strategy has not yet been determined.

Most recently, the COPEX search algorithm[44] proposed a systematic exploration of binary and fixed-composition phases using USPEX. Here, a ternary system is explored by (1) starting with simple binary phases (e.g., Y-C, Y-F, and C-F), then (2) moving to complex binary system searches where endpoints are stable binary phases (e.g., $(Y_2C)_x(YF_3)_{1-x}$), and finally (3) continuing with new binary phase links between stable phases (e.g., $(Y_2C)_x(Y_2CF_2)_{1-x}$). Fixed-composition and full-evolutionary ternary searches are carried out in tandem with this process but have a smaller influence on phase space exploration. To understanding this systematic algorithm, it helps to compare the exploration of a ternary diagram to the process of a spider creating its web (Figure 6-5a). Noting this analogy and that there are aptly named “random walk” search algorithms[45], we therefore refer to the COPEX algorithm as a “web walk” through candidate compositions, where nodes of the web are stable compositions.

An alternative systematic strategy can also be built off our binary search algorithm. In that algorithm, we run independent fixed-composition searches of increasing complexity. The same can be done for many-element systems. Because this would screen all possible compositions for a fixed site count before repeating for higher site counts, the algorithm can be visually compared to sonar scans of increasing quality (Figure 6-5b). We therefore refer to this search algorithm as “stepwise scanning”. To implement this algorithm successfully, we must accurately identify which compositions to skip or stop while the search progresses. This ensures that time is not spent in regions with high formation energies and no thermodynamically stable phases.

Other strategies can also be explored based on the target system and desired output. For example, if we only want the ternary structure with the highest formation energy and do not care for other stable phases, one can utilize linear search algorithms to rapidly find the desired composition.

In summary, we propose both web-walk and stepwise-scanning search algorithms as promising search strategies. It remains to be seen which of these will be most efficient and robust – or if another, unidentified algorithm would outperform these. However, for now, both algorithms should provide initial systematic means to explore complex chemical systems. The key factor remains: Simmate provides an efficient means of exploring individual compositions, making these higher-level searches possible.

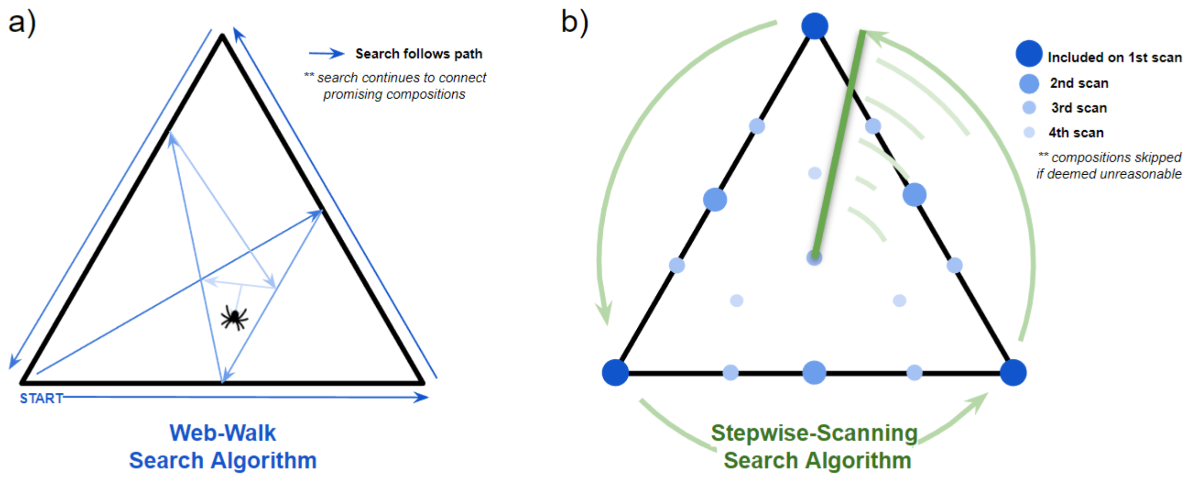

a)
Search follows path
** search continues to connect promising compositions
START
Web-Walk Search Algorithm
b)
Included on 1st scan
2nd scan
3rd scan
4th scan
** compositions skipped if deemed unreasonable
Stepwise-Scanning Search Algorithm

Figure 6-5. Illustration of web-walk and stepwise-scanning search algorithms. Searches for complex chemical systems and >2 element systems are possible within Simmate but have not yet been optimized for systematic exploration of the most promising subsystems. (a) Web-walk and (b) stepwise-scanning algorithms are two promising routes to explore complex phases efficiently, where a combination of these algorithms (via smaller parallel searches) can also be explored.

6.8 Benchmarking core components

Hitherto, we have presented our search algorithm assuming default settings, but we now turn to how our search was optimized and how further improvements can be made. We found that orders of magnitude improvement in search speed can be achieved through tuning of individual components, namely, structure creation, transformation, selection, and validation. There are diverse implementations of these components from many different software packages, and abstraction in the Simmate framework allows us to directly compare their performance. We therefore utilize Simmate's database framework for an informatics-driven analysis of core components.

6.8.1 Random structure creation

Structure creation is arguably the most important component of evolutionary structure prediction. Even if all other components work perfectly, a search may never find the ground state if the created structures do not identify promising regions of phase space. Moreover, extremely successful structure creation can negate the need for an evolutionary search altogether. At a minimum, structure creation must identify promising regions of phase space and rely on other components for local optimization and nearby phase exploration.

The most promising input structures come from existing data and experiments. This includes structures pulled from third-party databases, past calculations, and prototype libraries as well as probable substitutions of structures from these sources. We therefore attribute much of our search efficiency to its improved input seed structures and ability to pull from other similar search results. These sources do not, however, discover new and unexpected regions of phase space.

For broader phase space exploration, we must turn to random structure creation, which is challenging due to the number of possible structure configurations (section 6.3). We can only efficiently explore phase space by (1) limiting possible configurations to those most likely to be low-energy and (2) ensuring we sample many diverse regions of a phase space.

For (1), we can establish reasonable lattices and atomic distances. Given only a composition, we predict the total atomic volume by determining the most likely oxidation states for each element and then calculating total ionic volumes under hard sphere packing (at ~50% packing efficiency). When generating the lattice for this target volume, we choose to favor a more cubic and symmetric lattice. Therefore, vector lengths (a, b, c) are selected to be normally distributed in length, and angles (alpha, beta, gamma) are randomly selected from a range of 60°-120°. Lastly, atomic sites are placed randomly within the lattice under the condition that no sites are too close together – specifically, no closer than 75% of the summed ionic radii of two sites. Together, the resulting lattice and atomic positions provide a reasonable input structure, which can then be relaxed using the local optimization method (e.g., DFT or empirical force fields).

For (2), however, randomly created structures poorly sample phase space. On average, fully random structure creation leads to structures with similar unstable energies and glass-like amorphous ordering.[46] Converting these structures to higher symmetries via transformations is challenging and low probability, so it is better to improve structure crystallinity and symmetry during creation.

Structures are therefore created using random symmetry constraints from the 230 possible space groups. Lattice creation remains the same as discussed before, where vectors and angles are under the added constraints of the desired space group, e.g., $a=b=c$ and $\alpha=\beta=\gamma=90°$ for cubic systems. Meanwhile, the insertion of atomic sites is no longer straightforward. Rather than placing sites anywhere in the unit cell, sites are limited to the asymmetric unit for the corresponding space group – that is, the portion of the unit cell that can be used to produce the entire unit cell via the space group's symmetry operations. Depending on the atomic coordinates used for this asymmetric unit, a site may correspond to one or more atoms in the full unit cell after symmetry operations are applied. Each of these sites in the asymmetric unit are therefore represented by Wyckoff sites. For example, in a cubic cell, a new atomic site inserted at the Wyckoff site with coordinates (0,0, z) (z is between 0 and 1) will correspond to 3 symmetrically identical sites after symmetry is applied: (z, 0, 0), (0, z, 0), and (0,0, z). Because adding a single site in fact inserts multiple atoms, the selection of Wyckoff sites and their insertion must be carefully considered to prevent the creation of unrealistic structure, but there are multiple ways to approach this problem.

The first approach is a random walk strategy. Here, a random Wyckoff is selected, inserted into the asymmetric unit, and then checked for (1) site distances and (2) composition compatibility (i.e., the element counts are added to the desired composition after symmetry is applied). If the site passes these checks, a new site is inserted and evaluated. If the check fails, the site is removed, and a new attempt is made. If a new site repeatedly fails, the algorithm will take a step backwards and remove a previously accepted site. This process repeats until a valid structure with the desired composition and number of sites is produced.

The second approach utilizes the fact that there are a finite number of valid Wyckoff site combinations for the final structure. All possible Wyckoff sites are generated up front, and then a random combination is selected and used to generate all atomic sites at once. If atomic distance tolerances are not met, the Wyckoff combination generates new coordinates again until either a valid structure is successfully made or a maximum number of attempts is reached (and a new combination is selected). We refer to this as the random-symmetry-combo strategy.

The fully-random, random-symmetry-walk, and random-symmetry-combo approaches to random structure creation have been widely employed in software for evolutionary structure prediction. The following implementations have been integrated into Simmate and are compatible with our evolutionary search: ASE[8], PyXtal[47], XtalOpt[7] (RandSpg[32]), USPEX[6], CALYPSO[10], GASP[9], AIRSS[5], and custom Simmate implementation. ASE, GASP, and AIRSS rely on the fully-random algorithm; USPEX and CALYPSO implement the random-symmetry-walk; and PyXtal, XtalOpt, and Simmate use the random-symmetry-combo algorithm. Thus, we can gauge each algorithm's effectiveness by benchmarking all programs. We note, however, that lattice and atomic distance constraints frequently differ from Simmate's defaults that are described above.

We evaluated structure creation algorithms using the following compositions: Fe_1, Si_2, C_4, Ti_2O_4, Si_4O_8, Al_4O_6, $Si_4N_4O_2$, $Sr_4Si_4N_8$, and $Mg_4Si_4O_{12}$. Each creator was used to randomly create 500 structures, which is a relatively small sample size compared to the 230 possible space groups that can be selected. However, thanks to DFT geometry optimization, we found that identical structures often result from diverse input structures; in fact, we frequently see duplicate structures from < 10 atom compositions after relaxation. Thus, we find that a 500 structure sample size adequately represents the distribution of possible structures while also limiting the total computational time required. The 500 structures for each composition were used to evaluate a creator's (i) creation time, (ii) initial & relaxed structure similarities,

(iii) initial & relaxed energies, and (iv) CPU times for relaxation. The Simmate workflows for "*static-energy.vasp.quality04*" and "*relaxation.vasp.staged*" were used for energy determination and local optimization of all structures, respectively.

The average time required to create a new structure varies greatly between each program (Figure 6-6). This is because creation time is affected by many different factors: the programming language, the programmer's ability to implement the algorithm efficiently, the tolerances used to accept/discard structures, and the creation algorithm itself. Each of these components also affects how the program scales for larger structures. For example, all random-symmetry-combo creators (Simmate, PyXtal, and XtalOpt) each use the same algorithm but differ greatly in their speed and scalability due to different programming languages (Python for Simmate and PyXtal; C for XtalOpt), code quality, and tolerances. The results indicate that XtalOpt is the most efficient of these implementations, where the majority of computational time is attributed to Simmate's wrapper overhead for the XtalOpt library IO/EXT (e.g., time spent spawning a process for their program and reading output files back into Python). Furthermore, creation with GASP is extremely fast and scalable compared to other fully random algorithms, but this is likely due to their very forgiving tolerances for atomic distances and large lattice volumes. We therefore attribute diverse creation times to the many variables that also affect program speed.

We emphasize that creation times for all programs are negligible compared to local optimization times (discussed below), and these findings should not exclude any program from use in an evolutionary search algorithm. However, materials science is progressing toward machine-learned potentials for rapid local optimization, and there may be a point where creation times become rate-limiting and a much more important metric. But, until these potentials are utilized in evolutionary structure prediction, we do not need to discredit any program based on creation times alone.

Currently, it is more useful to evaluate structural diversity. Here, we use the CrystalNN fingerprint to calculate a statistical representation of a structure as a multi-dimensional vector, and the distribution of distances among all vectors is used as a measure of diversity.[48] Lower distances (i.e., more similar vectors) correspond to more similar structures.

The different creation algorithms can clearly be distinguished before and after structure relaxation (Figure 6-7). Perhaps counterintuitively, we see that fully random structure creators produce many similar structures with a limited diversity, which is in good agreement with previous reports.[29] Indeed, this result is expected because a fully-random structure is likely to generate a homogenous mixture, which is naturally similar to another homogenous mixture. In contrast, symmetry-based algorithms produce more diverse structures, but there is still a notable difference between random-symmetry-walk and random-symmetry-combo symmetry algorithms. Our findings indicate that the random-symmetry-combo algorithm produces a greater diversity in output structures and therefore a better sample of phase space. Furthermore, the advantage of the random-symmetry-combo algorithm is increased at more complex compositions because structural diversity decays more rapidly for fully random and random-symmetry walk. This indicates that a random-symmetry-combo is the optimal choice for phase space exploration, especially for structures with >10 atoms.

We can apply these findings to explain locally optimized structure energies as well. Ideally, lower energies are better, but only as long as the low energy structures are also structurally diverse. In the distribution of final energies, we can see that fully random algorithms such as ASE and GASP produce structures at lower energies on average, but the small distribution of energies (i.e., small 25-75% quartile ranges) indicates that we are arriving at similar structures. Combined with the lower fingerprint distances of these structures, we further corroborate the idea that fully random structure creation leads to many similar glassy/amorphous-like structures. The same is true for random-symmetry-walk structures, which produce lower-energy but less diverse structures on average.

These findings suggest that fully random and random-symmetry walk algorithms can still have an important role in evolutionary structure prediction. These algorithms may efficiently identify a promising region of phase space, while the random-symmetry-combo algorithm efficiently explores more diverse regions. One can introduce a small sample of fully random and random-symmetry walk structures at the start of the search (a single-shot source) and then prioritize random-symmetry combo structures as the search progresses (as a steady-state source). Thus, optimal search strategies may implement a combination of input structures from each of these different algorithms.

Energies prior to local optimization were excluded from the analysis above because we find that (a) initial energies are largely dependent on tolerances used during structure creation and (b) 500 structures insufficiently samples the distribution of possible structures. Nevertheless, we found that the distribution of energies for all programs and algorithms is surprisingly high, with a median energy above +0 eV/atom in some cases. This highlights how randomly generated structures are still poor estimates even when many tolerances and symmetry constraints are used. It also emphasizes how evolutionary structure prediction is heavily dependent on local optimization.

This highlights the importance of efficiently and robustly relaxing structures to local minima. These optimizations are the rate-limiting step for evolutionary searches, so any improvement in relaxation speed will have a profound effect. In fact, we found that relaxations of structures created by Simmate run an order of magnitude faster than those created by USPEX and two orders of magnitude faster than those created by CALYPSO (Figure 6-8). We attribute this to the improved starting point of each structure – via more accurate lattice volumes, atomic distance tolerances, and higher symmetry structures.

To confirm this, we increased the distance tolerances for structure creation and reevaluated the relaxation time. The stricter tolerance led to a halving of the median calculation time for structures and led to a small increase in structure diversity (Figure 6-9). We rationalize these results using two factors. First, these higher tolerances more accurately represent realistic, low energy structures, where lower energy structures take less time to geometrically optimize. Second, lower tolerances are analogous to high-energy molten salts, where optimization tends to yield more disordered, glassy phases – analogous to rapid quenching of a molten phase. Higher tolerances therefore lead to more diverse phases and faster relaxations.

The increase in relaxation speed and structural diversity are both desirable outcomes, and we propose that further optimization of these parameters can improve evolutionary search speed and throughput. Moreover, these tolerances can build off of existing search results to dynamically update these parameters as a search progresses.

In summary, significant progress has been made in the optimization of structure creation. We have evaluated key algorithms and programs and presented a new Simmate creator that produces diverse structures that are rapidly relaxed to local minima. We have also presented promising routes to further improve structure creation in the context of evolutionary search – specifically, utilizing a combination of input structures from different algorithms and dynamically updating creation tolerances as a search progress. Together, these findings will help accelerate the exploration and discovery of novel phases.

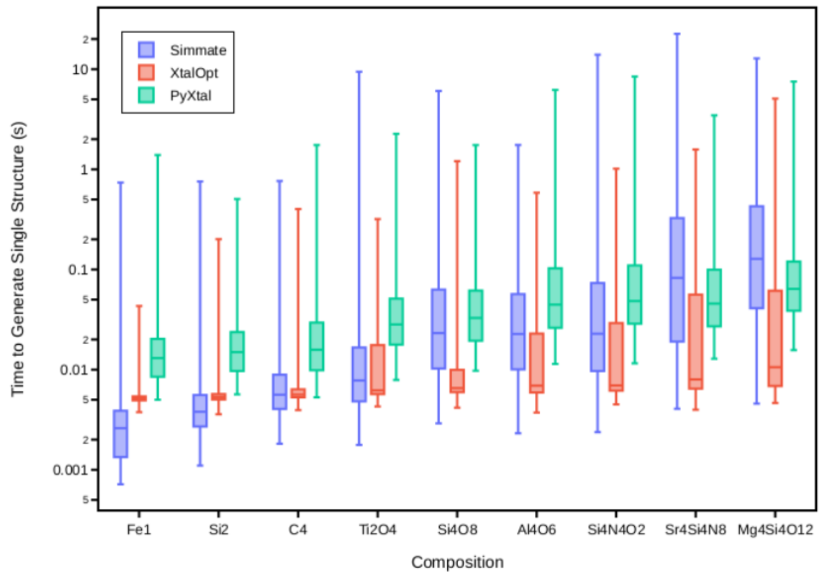


Figure 6-6. CPU time for random-symmetry-combo structure creators. Three different software packages are shown, and even though they implement an identical creation algorithm, there are many other factors (see main text) that affect creation speed and scalability.

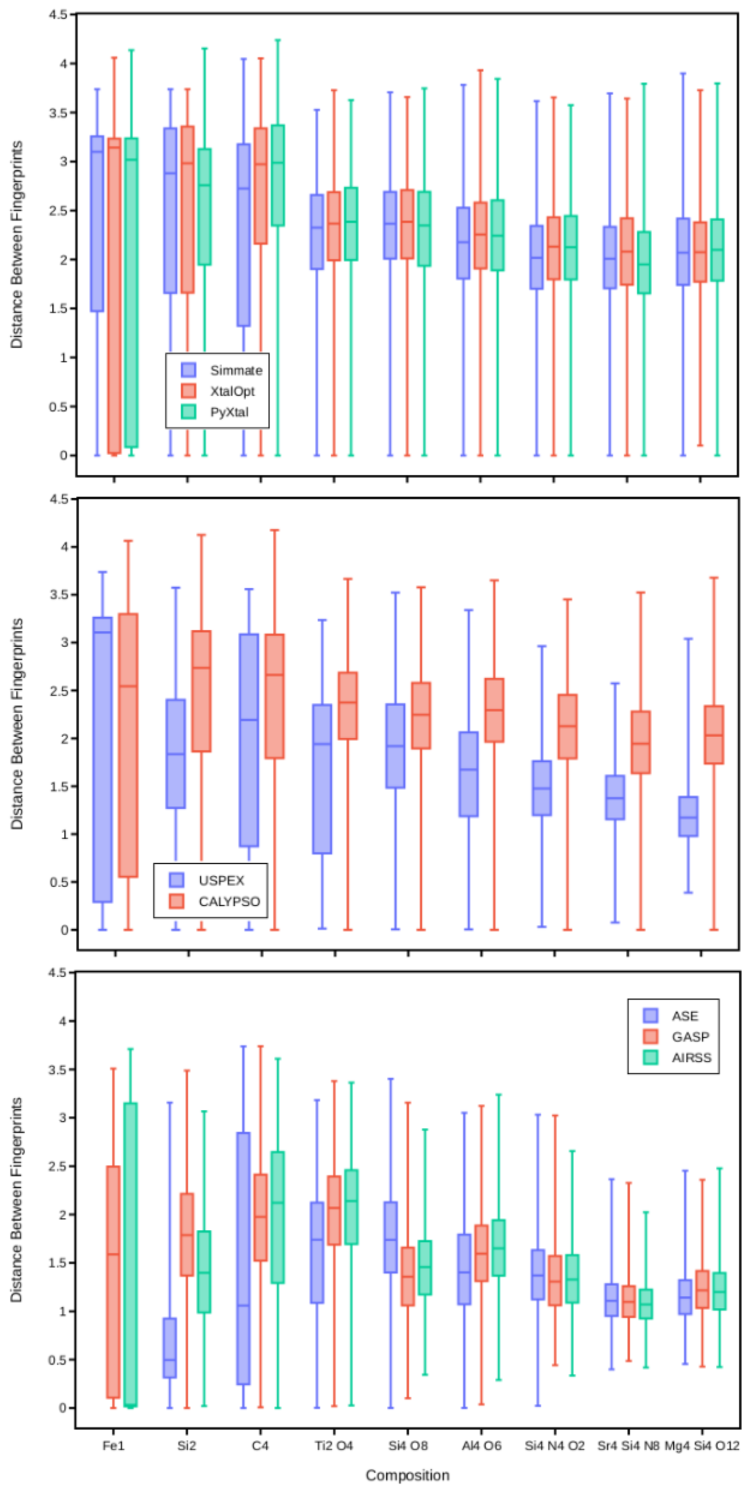


Figure 6-7. Structural similarities for different structure creation algorithms. There are three algorithms that are commonly used: random-symmetry-combo (top), random-symmetry-walk (middle), and fully-random (bottom).

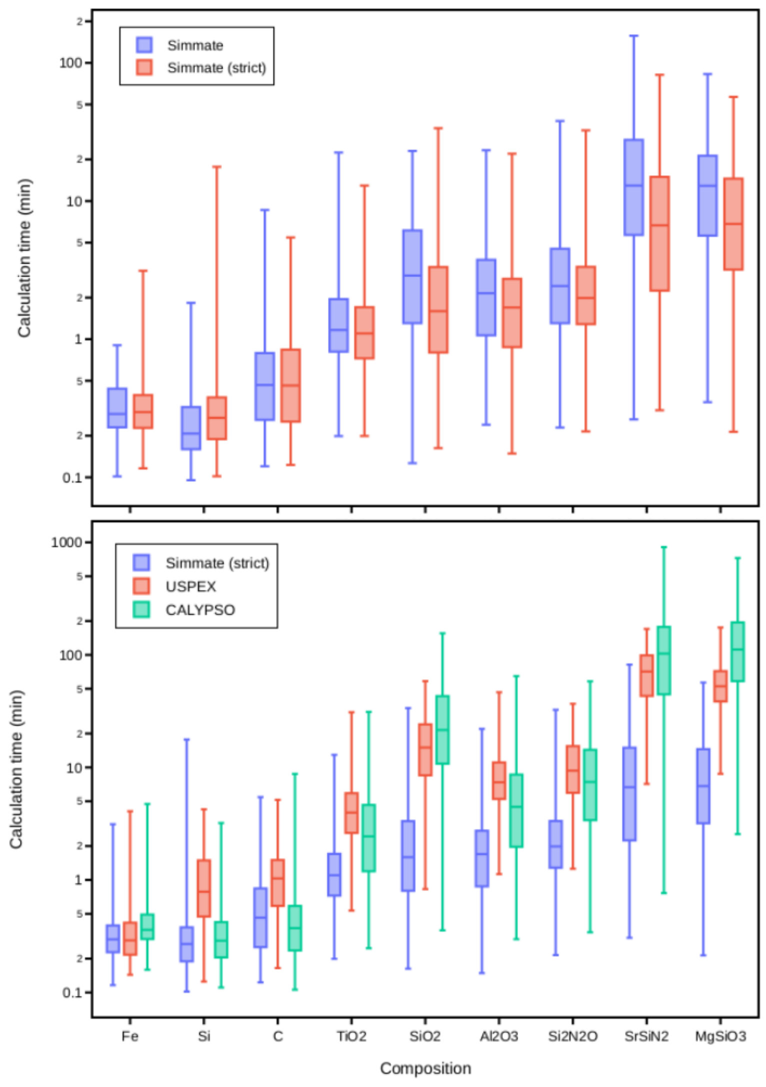


Figure 6-8. CPU times for local optimization of random structures. Simmate uses the random-symmetry-combo algorithm, which is highly sensitive to the tolerance values used (top). Stricter tolerances correspond to faster calculations, especially in larger systems. Optimization of Simmate's algorithm also leads to massive speedup relative to other structure prediction software (bottom).

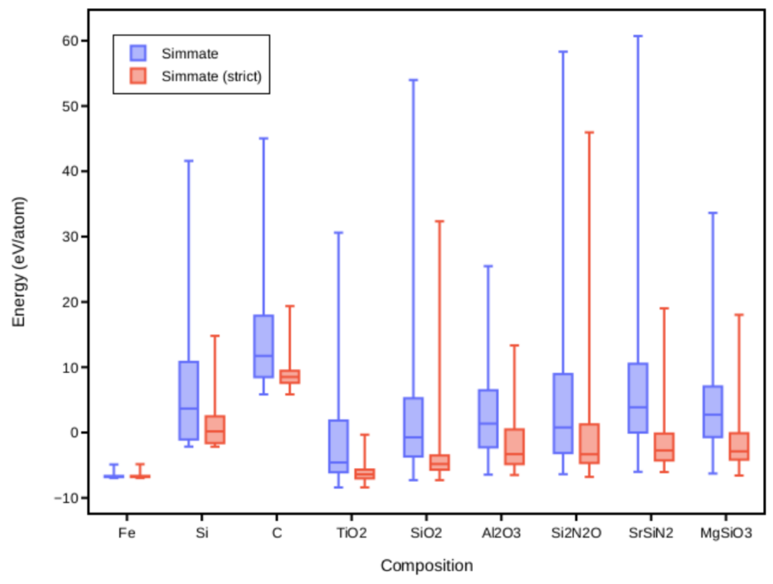


Figure 6-9. Final energies of random structures created under different distance cutoffs. Relative to the original values in our software (Simmate), lower-energy structures are created when atoms are required to be further apart (Simmate - strict).

6.8.2 Structure transformations

With so many possible structural transformations, it is important to evaluate the relative impact and success rate of each transformation. It is also likely that a transformation's relevance depends on the composition being tested or even how much the transformation has been used already. For example, one transformation may perform better on more disordered structures (i.e., higher atom counts) but should not be used until the search is largely converged (i.e., after >500 individuals). Steady-state sources and their relative proportions should therefore be dynamically determined and even change as the search progresses.

Rather than a predefined benchmark, transformations should instead be evaluated by analyzing search results over hundreds of compositional searches. The history of transformations can be used to determine the success rate ("success" = the transformation led to a lower energy structure than the parent structures) of each as well as how the probability of success varies across search stages and different compositions. Through this analysis, steady-state values can be optimized to favor the most successful transformations and do so at optimal stages of the search.

6.8.3 Parent structure selection

Many parent selection strategies have been developed for genetic algorithms, but Simmate currently implements only the most basic and common strategies.[49] Simmate uses tournament selection by default, where 20% of all individuals are randomly selected, and then the best individual of this random sample is used as a parent structure. Therefore, the lower energy an individual structure is, the more likely it will be chosen. Other selection methods operate under the same idea: best structures should have a higher probability of being selected to "reproduce". Truncated, roulette wheel, rank, or Boltzmann selection methods are among the others that could be applied.[49] There is no clear indication of which selection method performs the best in the context of structure prediction, so it is unclear how much an impact optimization will have here. What is known is that a mixture of parent selections and transformations is essential to drive a search forward: if selection and transformations are repeatedly generating identical structures, a search can become trapped in a local region of fingerprint space, making the search grossly inefficient.

6.8.4 Structure similarity and fingerprints

Despite the extreme number of possible crystal structures, a compositional search can often produces duplicate structures. This occurs because very different input structures can relax down to a common local minimum and, further, minor transformations of parent structures can lead to identical child structures. Identifying and removing these duplicate structures is important for optimal parent selection, interpreting results, and avoiding repeated calculations.

To address this, our search algorithm validates new structures by examining their fingerprints. Originally, the default fingerprint was the partial radial distribution function (pRDF), which captures the bond distance and atomic density of neighboring atom pairs. However, pRDFs are sensitive to small changes in structure volume, while we are more interested in removing duplicates that have matching coordinates. We therefore switched to the partial CrystalNN fingerprint[48], which quantifies the coordination environments for each element type. Initial testing shows that the frequency of duplicate structures is reduced relative to the pRDF fingerprint, but relative to pRDF, CrystalNN does not effectively capture structure arrangement beyond the first nearest neighbors. An optimal fingerprint that captures short-range coordination and long-range ordering is therefore needed if Simmate is to efficiently explore >20 atom compositions.

6.9 Future Outlook

6.9.1 Ability to collaborate and scale

The Simmate database and workflow framework can allow collaborative efforts at a scale that was previously not possible. This includes sharing computational resources and/or results.

Only an internet connection is needed to contribute, so resources across multiple universities and the globe can collaborate on highly parallel workflows. The only limit to the number of resources is from the database server itself (i.e., the maximum queries/connections it can accept at any time), but this can always be scaled to accommodate.

Importantly, it is also possible to share results and a database without sharing computational resources. This is a critical feature for some collaborations. Sharing computational resources can be costly and lead to security issues, whereas sharing only results is simple and often preferred. This can be done through labeling of computational resources, or in extreme cases, sharing of archive files instead of a shared database server.

Sharing data has important implications for accelerating projects and reducing the repetition of calculations. This is illustrated through the collaboration of similar or overlapping projects. For example, two researchers may be exploring the Y-C-F and Sc-C-F systems, where the C-F results can be shared and the similarity of scandium and yttrium can be used to generate promising seed structures. Without database sharing, each researcher would be running duplicate calculations on C-F and spend more time finding promising phases. Adding to this example, a third researcher could have a custom algorithm (completely separate from an evolutionary search) that explores promising fluoride-conducting materials from the Materials Project database. Here, if the researcher ran calculations on structures within the Y-C-F or Sc-C-F, then the other two researchers could automatically use those results in their search. Thus, diverse projects can share results – regardless of a project's motivation or priorities.

Therefore, one of Simmate's major goals is to get as many researchers to share a database as possible. As the user base grows, Simmate will facilitate new collaborations and avoid unnecessary repetition of calculations.

6.9.2 World's largest crystal dataset

Because each evolutionary search can result in tens of thousands of calculations, this work will result in the world's largest and most diverse database of high-quality structure energies and forces.

To illustrate the potential scale of this dataset, we can look at what resulted from our structure creator benchmarks. The benchmarks involved submitting a staged relaxation workflow across 7 different creators, 9 compositions for each creator, and 500 structures for each composition. The workflow submitted was a series of 5 relaxations and 1 static energy, so all creator benchmarks resulted in a total of 189,000 DFT calculations. When we explore what was automatically stored in the Simmate database, we find more than 4.5 million ionic steps and the data for each (energy, lattice stress, atomic forces). Moreover, all of these calculations were completed in one week and on a single HPC cluster (WarWulf, a 25-node cluster maintained by the Warren Lab).

The potential scale of the database becomes substantially larger when we consider searches for many chemical systems, the ability to scale across multiple HPC clusters, and the ability to load archives from contributors' searches. For example, over one week, we ran a ternary system search on Y-S-F, which involved ~120,000 total structures and a total of ~720,000 DFT calculations. This produced 11.75 million ionic steps, and all structures, energies, and atomic forces were stored in our cloud database. It is reasonable to expect this amount of data from each system search, and Simmate could easily exceed >1 billion ionic steps stored in a year. We have prepared for this scale of data by building a database server with the ability to extend storage capacity and by optimizing our database storage and archives. Simmate's database engine is therefore primed for an enormous scale of materials data.

The massive scale of this data becomes even more apparent when we compare it to existing database providers (Table 6-2). The Materials Project is the most widely used database, with only ~150,000 structures, while the AFLOW is the largest, with ~2.5 million structures. We anticipate Simmate will easily exceed these numbers in a year's time and, because Simmate is built to scale, we also include these other provider databases within ours by default. Furthermore, we expect to have significantly larger structural diversity due to the nature of evolutionary searches, while other databases are primarily limited to ground-state and experimental structures.

The Simmate database has been built to make the most out of all evolutionary search results. Evolutionary searches produce a large and diverse dataset of crystal structures, energies, and atomic forces, so the incorporation of a materials informatics framework makes the data analysis and distribution much more impactful. Simmate will make the data available through standardized APIs and web interfaces, allowing others to use the datasets for diverse applications.

Table 6-2. Scale of a database from evolutionary search results relative to other third-party databases supported by Simmate. Simmate's values are estimates based on our preliminary proof-of-concept calculations on the Y-S-F ternary system.

Provider	Number of Structures	Av. Sites per Structure	Archive Size
Simmate	>11 million per week	< 20	>2 GB per week
JARVIS	55,712	~10	8.0 MB
Materials Project	137,885	~30	45.2 MB
COD	471,664	~248	1.16 GB
OQMD	1,013,521	~7	79.2 MB
AFLOW	2,959,509	~5	1.06 GB

6.9.3 Accelerating nearby fields of material science

Our software and dataset can also serve to accelerate the modeling and understanding of structure-property relationships. This is largely due to the distribution of our large dataset to the community, which will promote diverse studies beyond just structure prediction.

For example, the dataset of energies and atomic forces can be used to accelerate the development of machine-learned potentials. Data mining and machine-learning studies require many atomic configurations to build accurate models, and Simmate search results can provide this input. Past studies have been successful using – most commonly – the Materials Project dataset or user-run molecular dynamics simulations. However, with the introduction of the Simmate dataset, the quality of these models may improve drastically and open up new applications.

Similarly, search results can be data mined to produce a spin-off dataset – such a new prototype library. Evolutionary searches are well positioned to identify new structure types for stable and metastable phases, and by going through all search results, one could contribute new template structures to the AFLOW prototype encyclopedia or even establish a new prototype library altogether. Not only would this facilitate the discovery of new materials, but it would also lead to a more complete ‘map’ of phase space and our understanding of phase stabilities.

Machine-learned potentials and prototype libraries are two obvious examples of how our dataset can be used, but many more avenues are possible. Ranging from property prediction to accelerated simulations, the possibilities will be grounded in the quantity and quality of data available. We aim to meet these needs through our material informatics framework and are excited to see how it enables the discovery of next-generation, high-performance materials.

REFERENCES

(1) DiSalvo, F. J. Solid-State Chemistry: A Rediscovered Chemical Frontier. *Science* **247**, 4943, 649–655. (1990).

(2) Sun, W. et al. The thermodynamic scale of inorganic crystalline metastability. *Sci. Adv.* **2**, 11, e1600225 (2016).

(3) Semieniuk, G., Taylor, L., Rezai, A. et al. Plausible energy demand patterns in a growing global economy with climate policy. *Nat. Clim. Chang.* **11**, 313–318 (2021).

(4) Oganov, A.R (Ed.). Modern Methods of Crystal Structure Prediction, *Wiley VCH* (2010).

(5) Pickard, C.J. & Needs, R.J. Ab initio random structure searching. *J. Phys.: Condens. Matter* **23**, 053201 (2011).

(6) Lyakhov, A.O. et al. New developments in evolutionary structure prediction algorithm USPEX. *Comp. Phys, Comm.* **184**, 4, 1172-1182 (2013).

(7) Falls, Z. et al. The XtalOpt Evolutionary Algorithm for Crystal Structure Prediction. *J. Phys. Chem. C* **125**, 3, 1601–1620 (2021).

(8) Vilhelmsen, L. B. & Hammer, B. A genetic algorithm for first principles global optimization of supported nano structures. *J. Chem. Phys.* **141**, 044711 (2014).

(9) Revard, B. C.; Tipton, W. W.; & Hennig, R. G. GASP: Genetic algorithm for structure and phase prediction. *Online Github Repository* https://github.com/henniggroup/GASP-python (2018).

(10) Wang, Y.; Lv, J.; Zhu, L.; & Ma, Y. CALYPSO: A method for crystal structure prediction. *Comp. Phys. Comm.* **183**, 10, 2063-2070 (2012).

(11) Jain, A. et al. Commentary: The Materials Project: A materials genome approach to accelerating materials innovation. *APL Mater.* **1**, 011002 (2013).

(12) Choudhary, K., Garrity, K.F., Reid, A.C.E. et al. The joint automated repository for various integrated simulations (JARVIS) for data-driven materials design. *npj Comput Mater* **6**, 173 (2020).

(13) Saal, J. E., Kirklin, S., Aykol, M., Meredig, B. & Wolverton, C. Materials design and discovery with high-throughput density functional theory: the open quantum materials database (OQMD). *JOM* **65**, 1501–1509 (2013).

(14) Curtarolo, S. et al. AFLOW: an automatic framework for high-throughput materials discovery. *Comp. Mater. Sci.* **58**, 218–226 (2012).

(15) Talirz, L., Kumbhar, S., Passaro, E. et al. Materials Cloud, a platform for open computational science. *Sci Data* **7**, 299 (2020)

(16) Gražulis, S. et al. Crystallography open database - an open-access collection of crystal structures. *J. Appl. Crystallogr*. **42**, 726–729 (2009).

(17) Sundberg, J.D; Benjamin, S.S.; McRae, L.M.; & Warren, S.C. Simmate: a framework for materials science. *JOSS* **7**, 75, 4364 (2022).

(18) McRae, L.M. et al. Sc2C, a 2D Semiconducting Electride. *J. Am. Chem. Soc.* **144**, 24, 10862-10869 (2022).

(19) Oganov A.R., Glass C.W. Crystal structure prediction using evolutionary algorithms: principles and applications. *J. Chem. Phys.* **124**, 244704 (2006).

(20) Mehl, M. J. et al. The AFLOW library of crystallographic prototypes: part 1. *Comp. Mater. Sci.* **136**, S1–S828 (2017).

(21) Hicks, D. et al. The AFLOW library of crystallographic prototypes: part 2. *Comp. Mater. Sci.* **161**, S1–S1011 (2019).

(22) Hicks, D. et al. The AFLOW library of crystallographic prototypes: part 3. *Comp. Mater. Sci.* **199**, 110450 (2021).

(23) Martoňák, R., Laio, A., Bernasconi, M., et al. Simulation of structural phase transitions by metadynamics. *Crys Mat* **220**, 5-6, 489-498 (2005).

(24) Martoňák, R., Laio, A., & Parrinello, M. Predicting Crystal Structures: The Parrinello-Rahman Method Revisited. *Phys. Rev. Lett.* **90**, 075503 (2003).

(25) Pannetier, J., Bassas-Alsina, J., Rodriguez-Carvajal, J. et al. Prediction of crystal structures from crystal chemistry rules by simulated annealing. *Nature* **346**, 343–345 (1990).

(26) Wales, D.J. & Doye, J.P.K. Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms. *J. Phys. Chem. A* **101**, 28, 5111–5116 (1997).

(27) Wang, Y., et al. Crystal structure prediction via particle-swarm optimization. *Phys. Rev. B* **82**, 094116 (2010).

(28) Oganov, A.R. Modern Methods of Crystal Structure Prediction. *Wiley-VCH* (2011).

(29) Glass, C.W., Oganov, A.R., & Hansen, N. USPEX—Evolutionary crystal structure prediction. *Comp. Phys. Comm.* **175**, 713-720 (2006).

(30) Ong, S. P. et al. Python Materials Genomics (pymatgen): A robust, open-source python library for materials analysis. *Comp. Mater. Sci.* **68**, 314–319 (2013).

(31) Hautier, G., Fischer, C., Ehrlacher, V., Jain, A., & Ceder, G. Data Mined Ionic Substitutions for the Discovery of New Compounds. *Inorg Chem* **50**, 2, 656-663 (2011).

(32) Avery, P. & Zurek, E.. RandSpg: An open-source program for generating atomistic crystal structures with specific spacegroups. *Comp Phys Comm* **213**, 00104655, 208-216 (2017).

(33) Ward, L., Dunn, A., Faghaninia, A., Zimmermann, N. E. R., Bajaj, S., Wang, Q., Montoya, J. H., Chen, J., Bystrom, K., Dylla, M., Chard, K., Asta, M., Persson, K., Snyder, G. J., Foster, I., Jain, A., Matminer: An open source toolkit for materials data mining. *Comput. Mater. Sci.* **152**, 60-69 (2018).

(34) Thompson, A. P. et al. LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales, *Comp Phys Comm* **271,** 10817 (2022).

(35) Kresse, G. & Furthmüller, J. Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Phys. Rev. B* **54**, 11169–11186 (1996).

(36) Giannozzi, P., et al. QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials. *J. Phys.: Condens. Matter* **21**, 395502 (2009).

(37) Gonze, X., et al. ABINIT: First-principles approach to material and nanosystem properties. *Comp Phys Comm* **180**, 12, 2582-2615 (2009).

(38) Lejaeghere, K., Speybroeck, V.V., Oost, G.V, & Cottenier, S. Error estimates for solid-state density-functional theory predictions: an overview by means of the ground-state elemental crystals, *Critical Reviews in Solid State and Materials Sciences* **39**, 1-24 (2014).

(39) Mathew, K. et al. Atomate: a high-level interface to generate, execute, and analyze computational materials science workflows. *Comp. Mater. Sci.* **139**, 140–152 (2017).

(40) Ong, S. P. et al. Custodian: A simple, robust and flexible just-in-time job management framework in Python. *Online Github repository* https://github.com/materialsproject/custodian (2022).

(41) Scott, E.O. & De Jong, K.A. Understanding Simple Asynchronous Evolutionary Algorithms. *FOGA* 85–98 (2015).

(42) Sundberg, J.D., Druffel, D.L., McRae, L.M. et al. High-throughput discovery of fluoride-ion conductors via a decoupled, dynamic, and iterative (DDI) framework. *npj Comput Mater* **8**, 106 (2022).

(43) Larsen, A. H. et al. The atomic simulation environment—a Python library for working with atoms. *J. Phys.: Condens. Matter* **29**, 273002 (2017).

(44) Liu, X., Niu, H. & Oganov, A.R. COPEX: co-evolutionary crystal structure prediction algorithm for complex systems. *npj Comput Mater* **7**, 199 (2021).

(45) Chupeau, M., Bénichou, O. & Voituriez, R. Cover times of random searches. *Nature Phys* **11**, 844–847 (2015).

(46) Oganov A.R., et al. Evolutionary crystal structure prediction as a method for the discovery of minerals and materials. *Rev. Mineral. Geochem.* **71**, 271-298 (2010).

(47) Fredericks, S., et al. PyXtal: A Python library for crystal structure generation and symmetry analysis. *Computer Physics Communications* **261**, 107810, 0010-4655 (2021).

(48) Pan, H., Ganose, A. M., Horton, et al. Benchmarking coordination number prediction algorithms on inorganic crystal structures. *Inorg Chem* **60**, 3, 1590-1603 (2021).

(49) Fortin, F.A., et al. DEAP: Evolutionary Algorithms Made Easy. *J Machine Learning Res* **13**, 2171—2175 (2012).