6 | Graph Generation via Adaptation

In drug discovery, lead optimization is the process of making slight changes to the structure of a candidate molecule (the lead compound) in order to improve its properties[*]. However, as a compound's structure is what determines its function, every structural change affects every property. A structural change that improves one property may harm another, and it isn't clear a priori which changes will produce the desired result. At present, this problem is approached with a combination of chemical intuition, expert knowledge, and a suite of virtual screening tools operating on cheminformatics libraries that can swap functional groups, atoms or bonds where physically plausible, exploring nearby compounds in chemical space. However, even with differentiable (neural) property predictors, because the enumeration procedure is not differentiable, optimizing for multiple properties in this model requires a guess-and-check approach. In this work, we take the first step to making this task end-to-end differentiable, transforming molecular property predictors into molecular manipulators. Specifically, we focus on a constrained variant of bioisosteric replacement, where we simply swap atom types within a given, fixed molecular structure. By absorbing the vectorized atomic features of a given input compound into a property

[*]I would like to thank Kevin Cusack for the productive, early discussions at AbbVie's MedChem Summit that set this project in motion.

predictor, we can optimize for them directly and locate new compounds with better properties and the same structure. Challenges remain, but we demonstrate that the method is capable of differentiably optimizing compounds towards multiple desired properties while preserving the original structure, a first in the bioisosteric transformation literature.

6.1. Methodology

Our goal is simple: to optimize a given lead compound towards multiple endpoints (properties) without changing its structure. Given this strong structural constraint, the task amounts to finding the optimal atomic distribution within that structure. We solve this problem using a novel methodology that allows us to transform existing molecular property predictors into graph generators without the need for retraining of any kind. This is a simple method for building a generative model from a predictive one.

6.2. Differentiable Property Predictors

These days, the best molecular property predictors are end-to-end differentiable. Our approach allows us to leverage them to intelligently navigate chemical space using the best guides available.

For maximum generality, we assume we already have a pre-trained molecular property predictor f_θ, parameterized by parameters θ, that maps molecular graphs onto a set of k properties[†]. Ordinarily, f_θ is a graph neural network, but it doesn't necessarily need to be. The only requirement on f_θ is that it be end-to-end differentiable and perform graph-level prediction, i.e., it maps entire graphs onto graph-level properties.

[†]In our experiments, this was a property predictor developed at AbbVie, but our methodology applies equally using any open source model so long as it is end-to-end differentiable.

We denote the graph corresponding to a single molecule with $G(V, \mathcal{E}) : V \in \mathbb{R}^{N \times d}$, where N is the number of atoms in the molecule, d the number of features per atom, and $\mathcal{E}$ the set of bonds joining adjacent atoms. The predictor f_θ takes this graph as input and maps it onto each of k molecular properties as follows:

$$f_\theta\big(G(V, \mathcal{E})\big) = y,\ y \in \mathbb{R}^{k \times 1} \tag{6.1}$$

6.3. Neural Atomic Replacement

Here, we introduce our methodology, termed Neural Atomic Replacement (NAR). Bioisosteric Replacement is simply the task of swapping functional groups or, in our case, atoms in a molecule, with the goal of improving some molecular properties. Our method is the first learnable, differentiable solution to this atom-swapping problem.

To use this trained model to intelligently modify an input graph, we are going to recast the input features $G(V, \mathcal{E})$ - specifically the node features $V \in \mathbb{R}^{N \times d}$ - as *parameters* of a new, composite model containing both the trained model f_θ and the molecular graph $G(V, \mathcal{E})$. We denote our composite model, $h_{f_\theta, V, \mathcal{E}}(\cdot)$. The composite model h is parameterized by both the trained parameters θ *and* the features of an input compound V. The output of h then becomes:

$$h_{f_\theta, V, \mathcal{E}}(\cdot) = f_\theta\big(G(V, \mathcal{E})\big) = y, y \in \mathbb{R}^{k \times 1} \tag{6.2}$$

Note that Unlike f, h does not take any input (it is implicitly encoded into the model) but still produces a vector of predicted property values $y \in \mathbb{R}^{k \times 1}$.

6.3.1. Optimization

Now that we've built our composite model, we can use the standard machinery of back-propagation to optimize the input graph G.

To do so, we need targets. However, these targets won't be the standard labels provided in $\hat{y}$, but rather the *desired* values for each property. When designing a drug compound, chemists often have a desired property profile in mind. In order for drugs to be effective, they first need to be absorbed, distributed, metabolized, and excreted (ADME) at certain rates. These ADME properties are the properties that the predictor f_θ yields. When optimizing the input graph using the composite model, we denote the desired values for each property with $\hat{y}_{opt}$. From here, updating the features of the input graph becomes trivial: we perform a single forward step with h to generate predicted values y, calculate the loss between y and the *desired* values $\hat{y}_{opt}$ using a loss function $\mathcal{L}(\cdot)$. Then we calculate the gradients $\frac{\partial \mathcal{L}}{\partial V}$ which are used to update only the parameters V in $h_{f_\theta,V,\mathcal{E}}$.

This solves the problem we're after: it modifies the qualities of the *input graph* (drug molecule) in such a way as to push the *predicted* property values towards each of the desired values.

However, given the physical constraints of organic chemistry, this process alone is not sufficient to yield a usable generative model. We must also respect the space of viable atoms, i.e., which atoms are usable in an organic drug compound while maintaining the differentiability of the model.

6.3.2. Fragmenting Atomic Space

One detail we have not mentioned yet is what each of the node features in V represents. In the case of molecular graphs, we make use of the widely used RDKit framework to preprocess the input molecules and generate their graphical representation. For each atom in the molecule, nine features are generated; these features are:

(1) Atomic Number: Identifies the element of the atom.
(2) Chirality: Describes the three-dimensional arrangement of the atom, particularly in stereochemistry.
(3) Degree: The number of explicit bonds the atom has with other atoms.
(4) Formal Charge: The electric charge of the atom.
(5) Hybridization: The type of orbital hybridization (e.g., sp, sp2, sp3).
(6) Implicit Valence: The number of implicit hydrogen atoms or other single-bonded atoms connected to the atom.
(7) Isotope: The specific isotope of the element (if applicable).
(8) Number of Radical Electrons: Electrons that are not paired.
(9) Aromaticity: Whether the atom is part of an aromatic system.

The feature we are most concerned with is the first, atomic number. This defines the type of atom at each position and is the only feature that we update in the optimization process. This is a limitation and will be discussed in a later section but, as our results demonstrate, even changing only this value is sufficient to generate novel compounds with properties closer to the desired properties set forth in $\hat{y}_{opt}$.

The challenge is as follows: using the backprop machinery described above, we will iteratively make small changes to the values in V, gradually shifting the compound in 'atom-type space' towards a compound with more desirable properties. For example, if an atom starts out as a Carbon but a Nitrogen there would yield better properties, we would gradually shift that value from 6.0 to 7.0, the atomic numbers for Carbon and Nitrogen, respectively.

However, atoms can only have one integer atomic number or another, not something in between. So, we need a way to bridge the gap between atom types in feature space without quantizing and breaking the differentiability. To do so, we introduce the concept of feature-space fragmentation.

As an additional note, only some atom types are usable in organic chemistry. From all the *possible* atom types, we select only C, N, O, F, S, and Cl, or atomic numbers 6, 7, 8, 9, 16, and 17[‡]. To span the gulf between the first cluster of atom types 6 - 16 and the second 16 - 17, while maintaining differentiability and allowing backpropagation to effectively optimize this feature, we introduce the atomic-space fragmentation scheme in Figure 6.1.

First, we don't allow the features in V to take on values between these two blocks, shown by the "eliminate gaps" transition in Figure 6.1. Second, we localize the possible values around each of the allowed atom types, shown in the "localize atoms" transition.

[‡]This is mostly due to synthetic constraints.

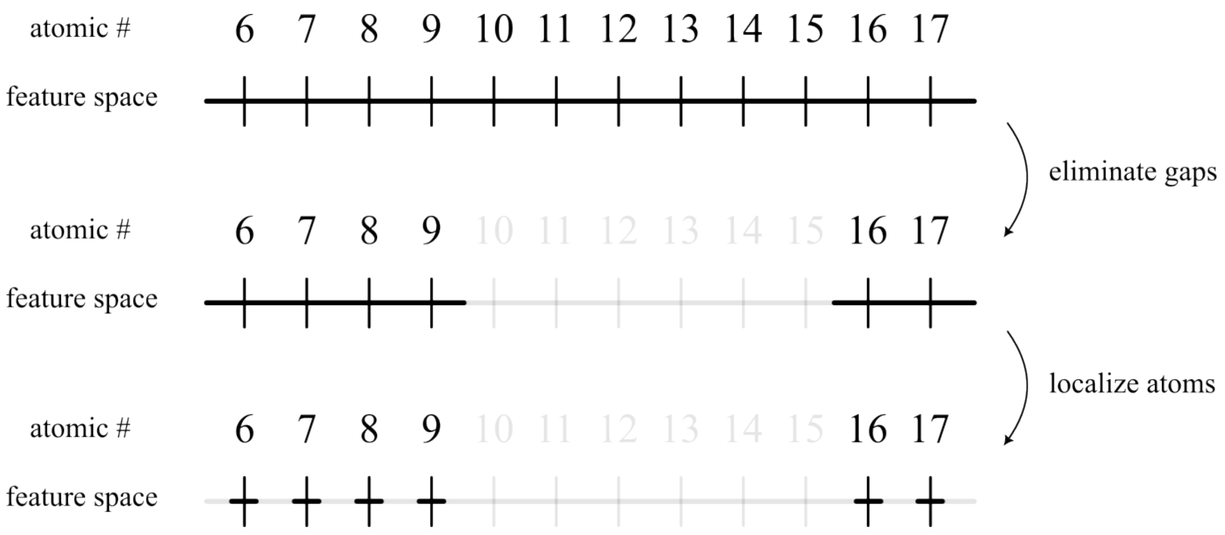

Figure 6.1. The atomic-space fragmentation scheme visualized. At the top, we show the entire space spanning both clusters of allowable atoms. Our first step is to eliminate the space between the two clusters. Then, to further improve the efficiency and performance of the generative scheme, we also localize the allowable feature space around each atom. For example, if we provide an allowable window of width 0.5, the first two allowable windows would be [5.75, 6.25] and [6.75, 7.25]. This window size is a hyperparameter and may vary based on the application.

The width of each of these windows is a hyperparameter, but we have found that a width of 0.5 centered at each atom type yields good performance.

Now, when we are optimizing the atomic numbers in V using the method defined above, we will only allow those features to take on values in the black regions in Figure 6.1, which brings us to our next methodological development: Feature Value Snapping.

6.3.3. Feature Value Snapping

Feature values in V are iteratively modified using the standard machinery of backpropagation. If such a feature value falls into one of the disallowed regions (shown in gray in Figure 6.1, we perform what we term *Feature Value Snapping.* Assume, for example, that we have two windows 1) [5.75, 6.25] corresponding to Carbon, and 2) [6.75, 7.25] corresponding to Nitrogen, and that a feature value in V was adjusted from 6.25 to 6.35 using a learning rate of 0.1. Instead of keeping the value 6.35, which is not in an allowed region, we would snap it up to the bottom of the next window by updating its value to 6.75, placing it into the allowable window for atomic number 7. This procedure accelerates the optimization process by allowing the model to spend less time in the spaces between atom types, and we also find that it encourages the model to learn more optimal solutions.

6.4. Experiments and Results

In our experiments, we set out to answer the following questions:

Q1 - Can our method generate novel and valid compounds with identical structures to the seed compound?

Q2 - If so, do those compounds have more desirable properties than their seed compound? As defined by the targets we set.

Q3 - Further, can our method locate compounds that improve upon multiple competing properties simultaneously?

To find out, we've gathered a random sample of ~1,000 compounds from the Therapeutic Data Commons [**50**]. For each of these seed compounds, we generate a *single* new compound using our method. This new compound has an identical structure, (possibly) different atoms at each position, and is optimized towards one, two, three, or four ADME properties simultaneously. We've chosen one assay from each of the four ADME categories - Absorption, Distribution, Metabolism, and Excretion - to make the experiments as representative as possible.

Joint Targets	A	D	M	E
	Caco2	VDSS	Fu Mic	Clearance
A	+78.2%	-	-	-
A&D	+70.6%	+23.6%	-	-
A&D&M	+71.2%	+20.2%	+54.5%	-
A&D&M&E	-16.3%	+57.1%	+6.81%	+70.1%

Table 6.1. Median improvements to each of five endpoints (graph properties) our method was able to achieve for property A by itself, A and B, A and B and C, etc. A value of +10% means that the predicted property value of the generated compound was 10% closer to the desired value than the predicted property value of the seed compound. Higher is better, and the uniformly positive performance shows that our method is able to robustly generate new compounds with identical structures but better properties than their seed compounds. Further, rows 2-4 show settings in which we are attempting to optimize for multiple properties simultaneously. This table shows that our method is capable of generating new compounds that improve upon multiple competing properties at once.

Our results are shown in Table 6.1, which presents the performance of the modified compounds when optimizing toward one, two, three, or four of the ADME endpoints. We report the median performance to avoid results being skewed by outliers, which are common in this space. Results are expressed in terms of *improvement* in the predicted property value. For example, a value of +10% means that the predicted property value of the generated compound was 10% closer to the desired value than the predicted property value of the seed compound. This is what we are aiming for and what we achieve.

To answer the questions set forth above:

A1 - Yes. our method generated a novel and valid compound for each of the seed compounds tested.

A2 - Yes. Predicted property values for the generated compounds uniformly improved upon the seed compounds. The only scenario that saw performance degradation was when optimizing for all five endpoints simultaneously, where the Absorption metric decreased. This is due to a well-known competitive interaction between absorption and clearance that we will discuss in Sec. 6.7.

A3 - Yes. Up to three endpoints, our method was capable of improving upon all three simultaneously.

Table 6.1 clearly demonstrates the utility of the method, and we are proud to say that this has been used at AbbVie in the lead optimization phase and generated a compound sufficiently compelling as to be physically synthesized and tested. A significant milestone for any generative method.

6.5. Methodological Details

Given a trained property predictor f_θ, our method has relatively few hyperparameters. They include:

- Learning rate: the size of the changes made to each feature value per iteration
- Number of iterations: how long we run the adaptation/optimization procedure
- Endpoints: the particular properties we're aiming to optimize
- Feature-space fragmentation: how (and if) we decide to fragment the feature space to improve generative efficiency

In our experiments, we choose a learning rate of 0.1. While typically too large a learning rate to train a neural network, we've found that this larger rate allows the method to explore further in feature space and find more optimal solutions.

We choose to optimize for 150 iterations, as this balanced performance with runtime. We note, however, that performance increases the longer we run.

The endpoints we chose were for demonstration purposes only. Individual projects will have their own set of properties they're optimizing for and our method can accommodate these without issue.

In regards to feature space fragmentation, we chose a window of 0.5 because it worked well and was easy to implement. Smaller windows may further improve performance and this is an important direction for future research. There is also the possibility of snapping to the middle of the next window, as opposed to snapping simply to the next edge. We will explore these options in future experiments.

We also note that at the end of the training, modified feature values in V are not integers, but atom types are. As such, we round them to the nearest integer before performing the final inference and assessing the quality of that generated compound.

6.6. Challenges

Many challenges remain, particularly as it relates to incorporating chemical rules into the method. Due to valency constraints, not just any atom can be placed in any location in an atom. At present, we handle this on an atom-by-atom basis, checking the valency of the proposed atom at that position and only allowing it to be swapped into the molecule if that constraint is satisfied. This is inefficient, and a more mature implementation would take this into account in the generation phase.

Atom type is also inherently a categorical value, and our treatment as a continuous value is only to maintain differentiability in this example. Future work could be directed at ways of handling categorical features in a more typical manner.

There is also the challenge of computational efficiency. Similar to diffusion, we need multiple iterations to modify the compound and converge onto a solution. This is an unavoidable property of the method as-is but does not represent a serious detriment because of how the method is intended to be used: to optimize select compounds one at a time. As this is not intended to be used to generate large numbers of compounds for screening, this limitation may be less significant than in diffusion.

Finally, as the property predictor f_θ is fixed, our method is deterministic given a single input molecule. One way around this that we are exploring is to simply add noise to the generation process. Adding noise on a preset schedule could help the model explore more possibilities early on in the optimization while also allowing for the identification of multiple modified compounds for each seed compound.

6.7. Discussion

Beyond just performant, our method is also targetable, meaning that we can trivially target only a given subgraph or functional group by masking the gradients applied to the input features. This is useful in the many cases where chemists know that some part of a compound is important for their project and want to it to remain fixed. In a similar vein, selectively fragmenting the feature space allows us to explore only particular chemical spaces by restricting which atom types we consider. This is useful when accounting for the global demands of organic chemistry or the local demands of particular retrosynthesis capabilities.

The fourth row of Table 6.1 shows clearly the expected competition between absorption and excretion. Generally, if a drug is absorbed more efficiently into the bloodstream, it is available in the body for a longer duration before being excreted. Conversely, if a drug is rapidly excreted, its absorption into the bloodstream may be reduced.

Finally, we are proud to say that, through our collaboration with AbbVie, this method has been used in lead optimization for an active drug discovery project and generated an optimized compound that has been physically synthesized and tested. This is a significant achievement, inducting our method into the small cohort of machine learning approaches that have yielded compounds worthy of synthesis.