ExaLearn-Design: Computational Design at Exascale with Deep Reinforcement Learning on Graphs

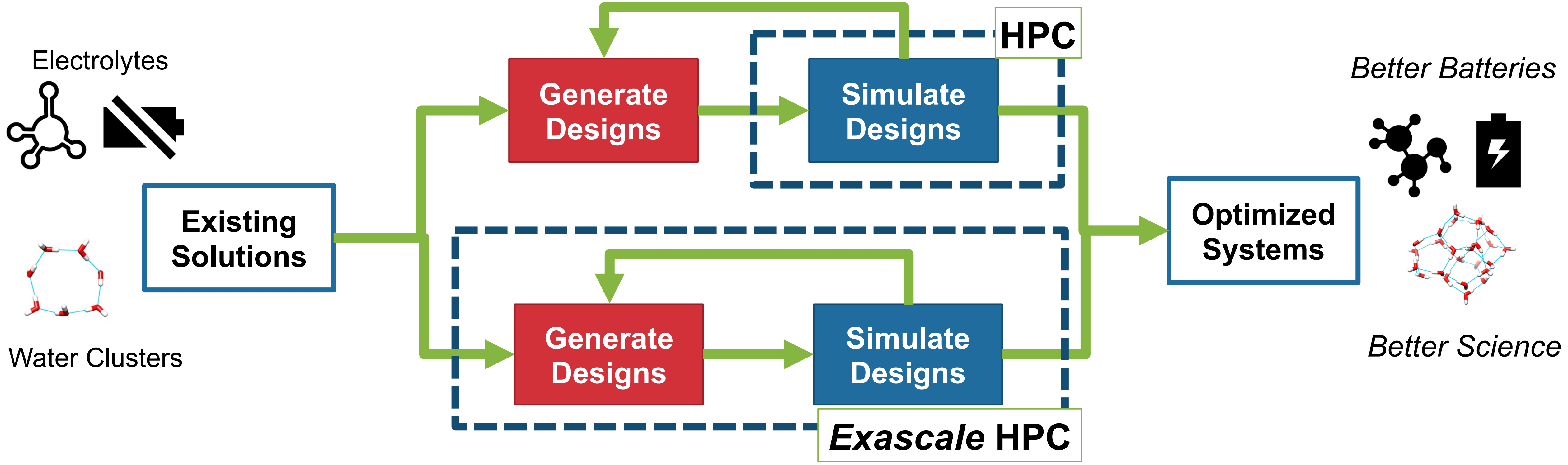
EXASCALE COMPUTING PROJECT

Sutanay Choudhury (PNNL), Logan Ward (ANL), Jenna Pope (PNNL), Malachi Schram (PNNL), Joseph Heindel (University of Washington), Sotiris Xantheas (PNNL), Marcus Schwarting (ANL), Yinzhi Huang (PNNL), Sayan Ghosh (PNNL), Jim Ang (PNNL), Ian Foster (ANL)

National Nuclear Security Administration

U.S. DEPARTMENT OF ENERGY | Office of Science

Design: Expanding Computational Design to the ExaScale

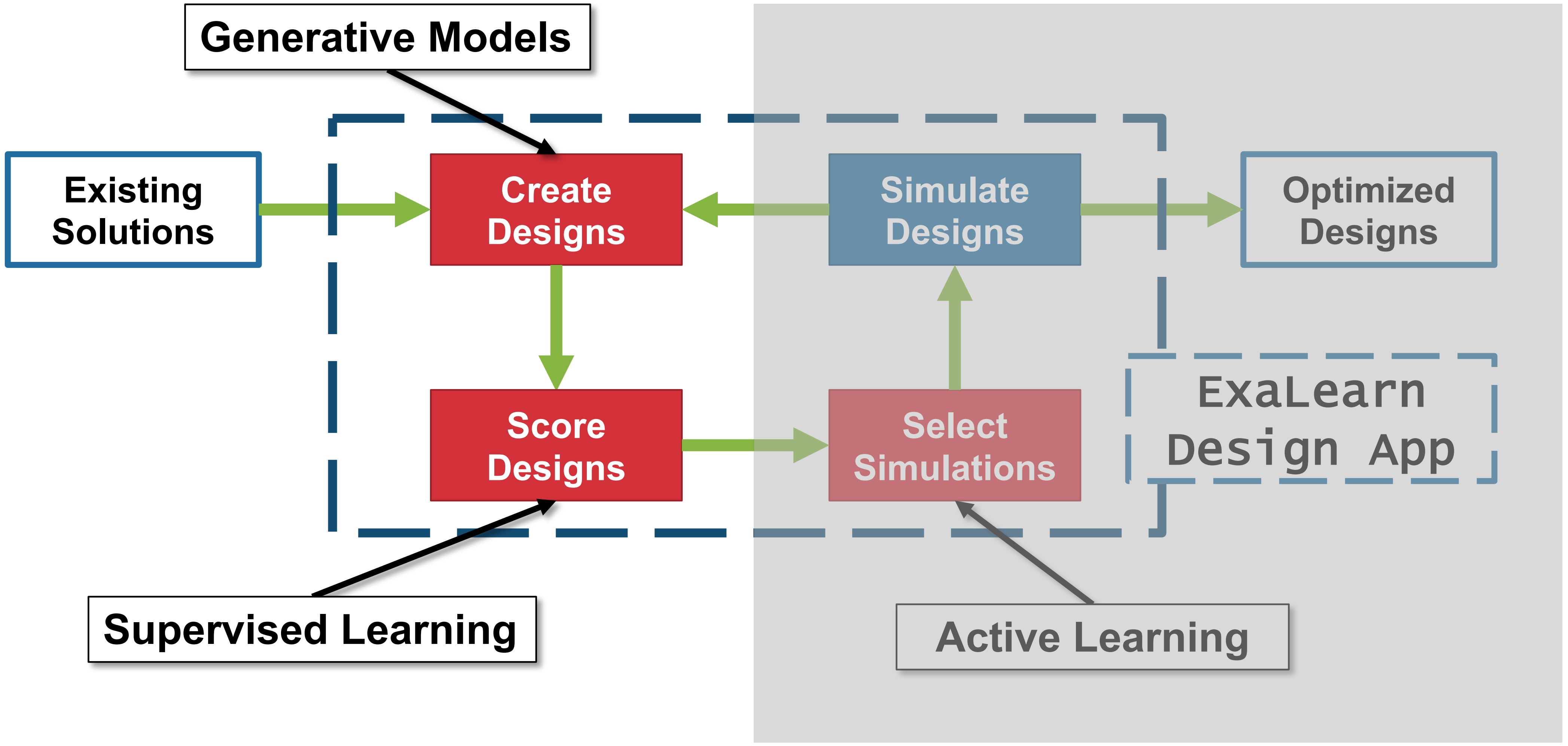


Today: Humans steer HPC, HPC performs simulations **Needed Solution:** HPC steering itself

Why Exascale?

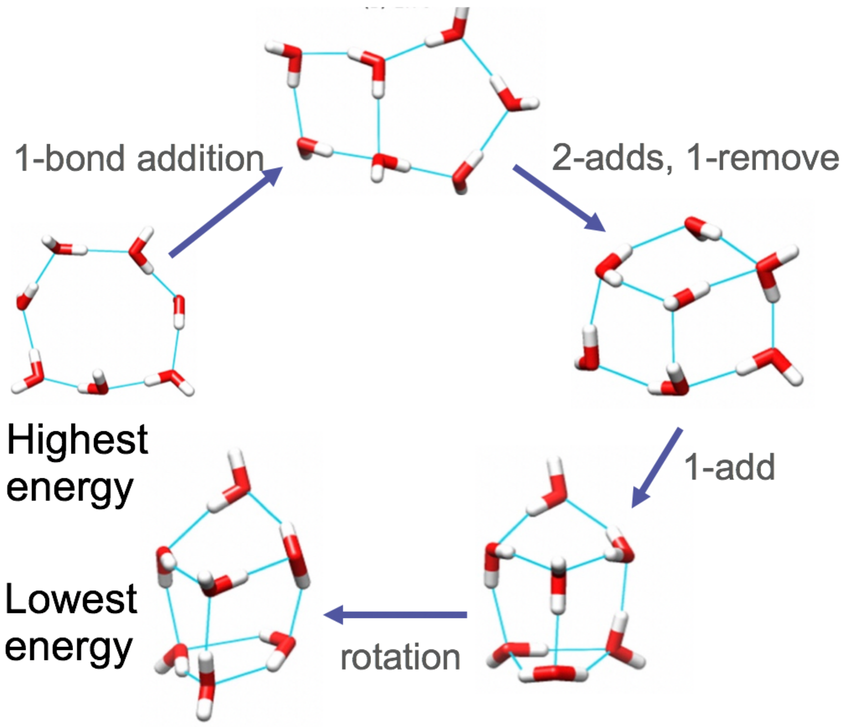
Large computational cost (>10^{11} molecules in GDB17)
Tight coupling between heterogeneous computations

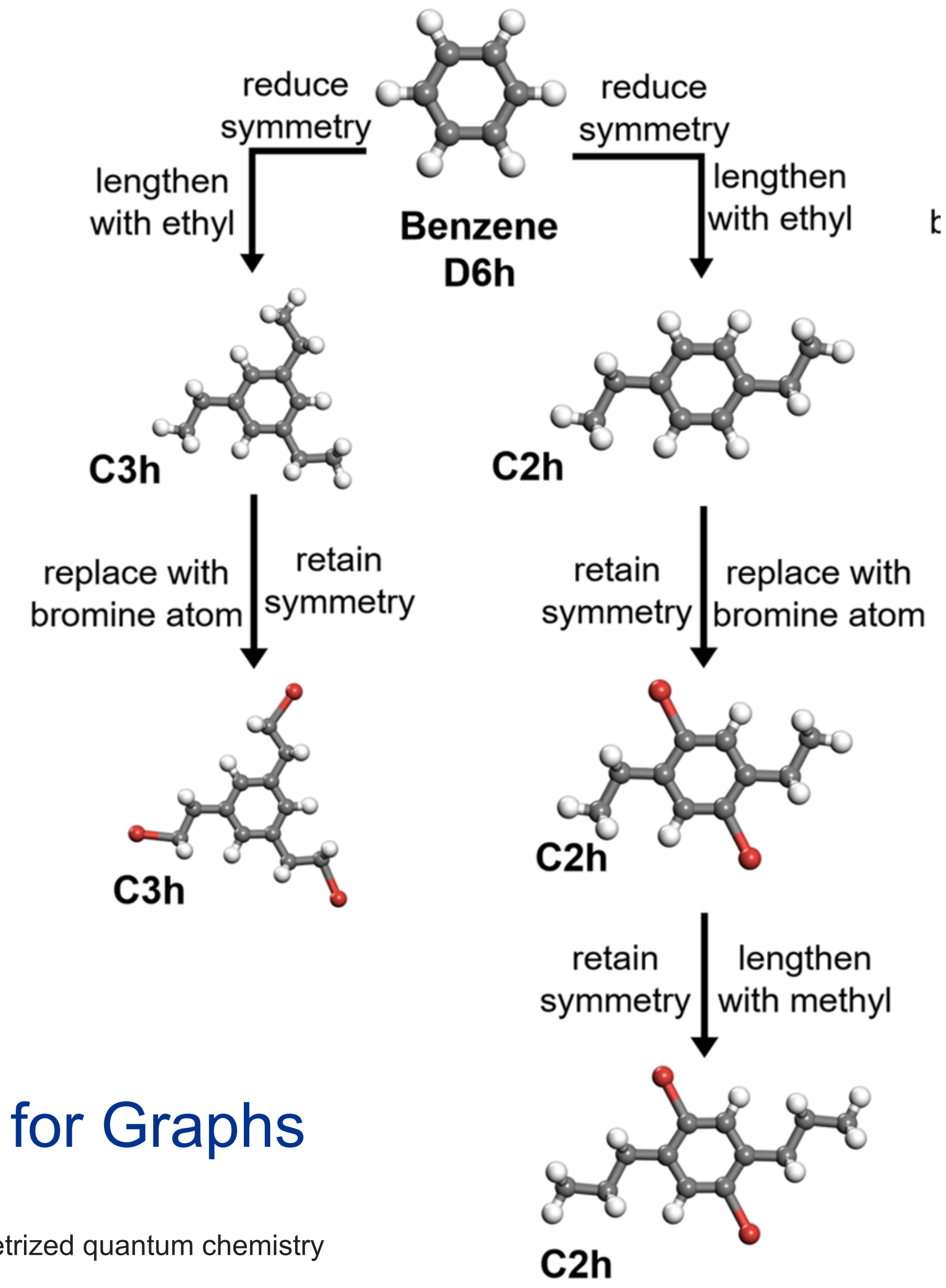
HPC Steering HPC Requires Extensive Machine Learning



Motivation

How do we automate the design of chemical structures that have interesting properties?



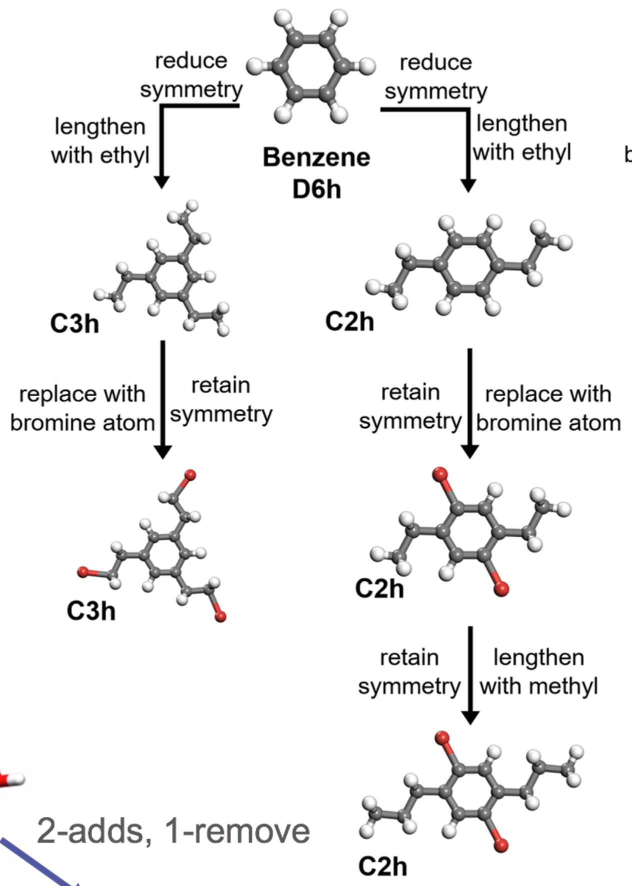


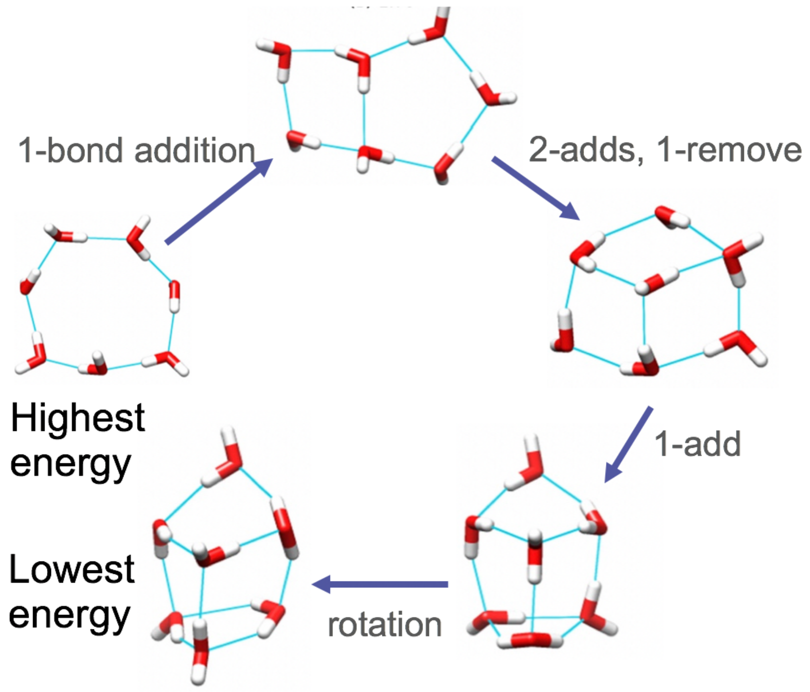
Our approach: Deep Reinforcement Learning for Graphs

Right figure from: Liang, J., Xu, Y., Liu, R. and Zhu, X., 2019. QM-sym, a symmetrized quantum chemistry database of 135 kilo molecules. *Scientific Data*, *6*(1), pp.1-8.

Tutorial Objectives

- Formulating Design as a Graph Reinforcement Learning Problem
- Training Graph Surrogate Models at Extreme Scale
- Developing Reward Functions via Graph-Theoretic Chemical Descriptors

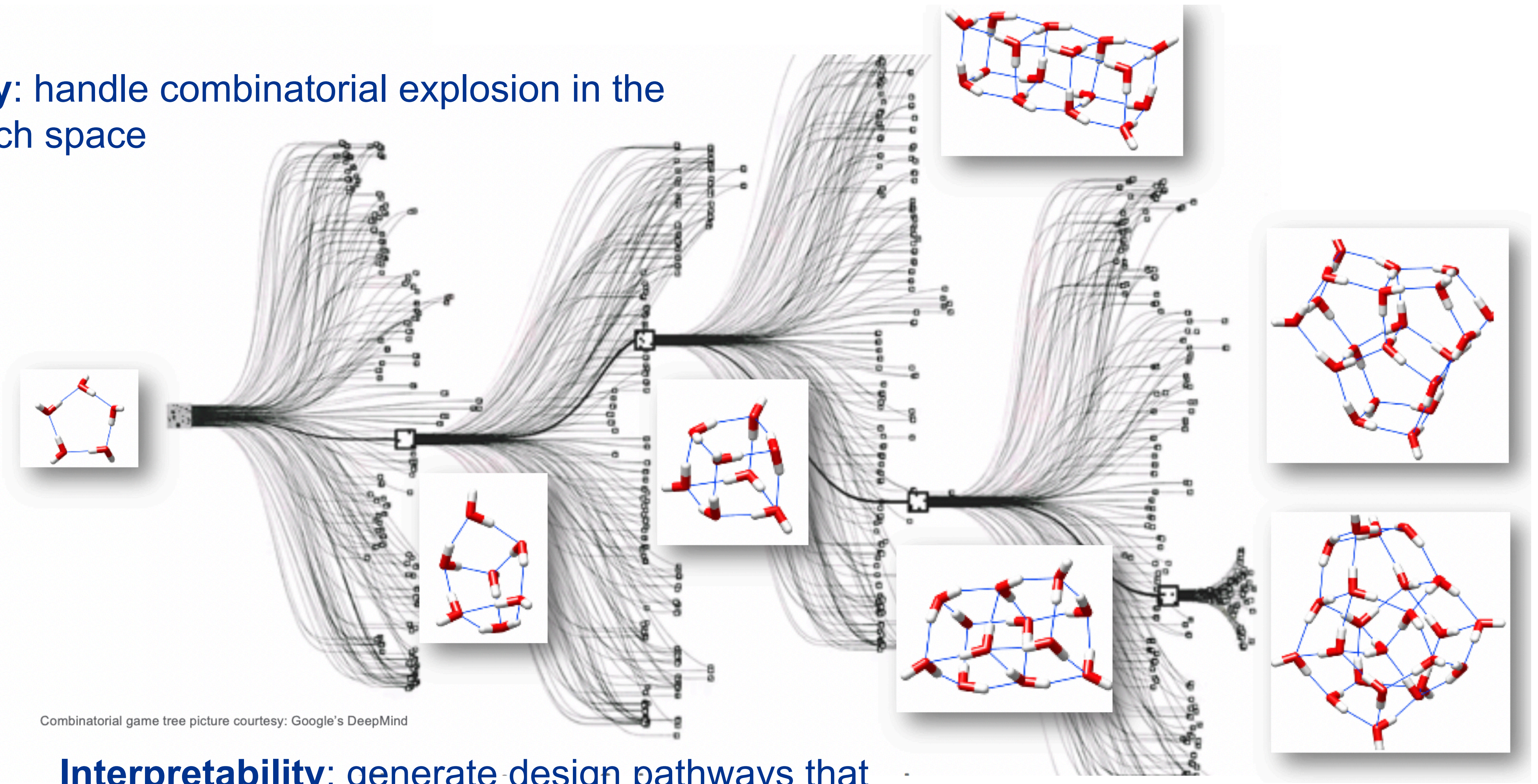




Formulating Design as a Graph Reinforcement Learning Problem

Key Goals: Scalability and Interpretability

Scalability: handle combinatorial explosion in the state-search space



Combinatorial game tree picture courtesy: Google's DeepMind

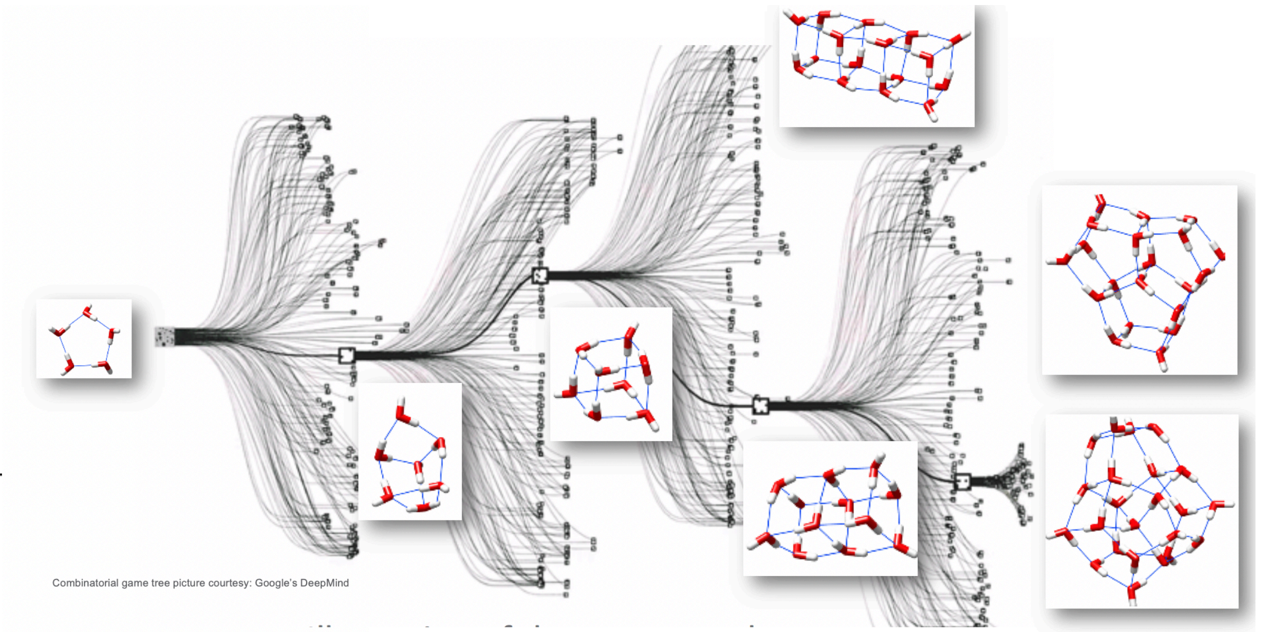
Interpretability: generate design pathways that scientists can reason about

How will we find our Target Structures?

Algorithm 1 A search algorithm to find a target molecular structure starting from an initial structure G_{mol}^{init} using a reinforcement learning model M_{RL} and branching factor b.

```
1:  procedure SEARCH($G_{mol}^{init}, b, M_{RL}$)
2:      INIT($results, \emptyset$)
3:      INIT($\mathbb{P}_{cand}$, EXPLORE-NEXT($M_{RL}, [G_{mol}^{init}], b$)))
4:      while $size(\mathbb{P}_{cand}) > 0$ do
5:          $pathway$ = POP($\mathbb{P}_{cand}$)
6:          $N_{cand}$ = EXPLORE-NEXT($M_{RL}, pathway, b$)
7:          for all $G_{new} \in N_{cand}$ do
8:              $P_{new} = pathway \cup \{G_{new}\}$
9:              if MATCH-TARGET($G_{new}$) then
10:                 $results = results \cup \{G_{new}\}$
11:             else
12:                 PUSH($\mathbb{P}_{cand}, P_{new}$)
13:     RETURN $results$
14: end procedure
```

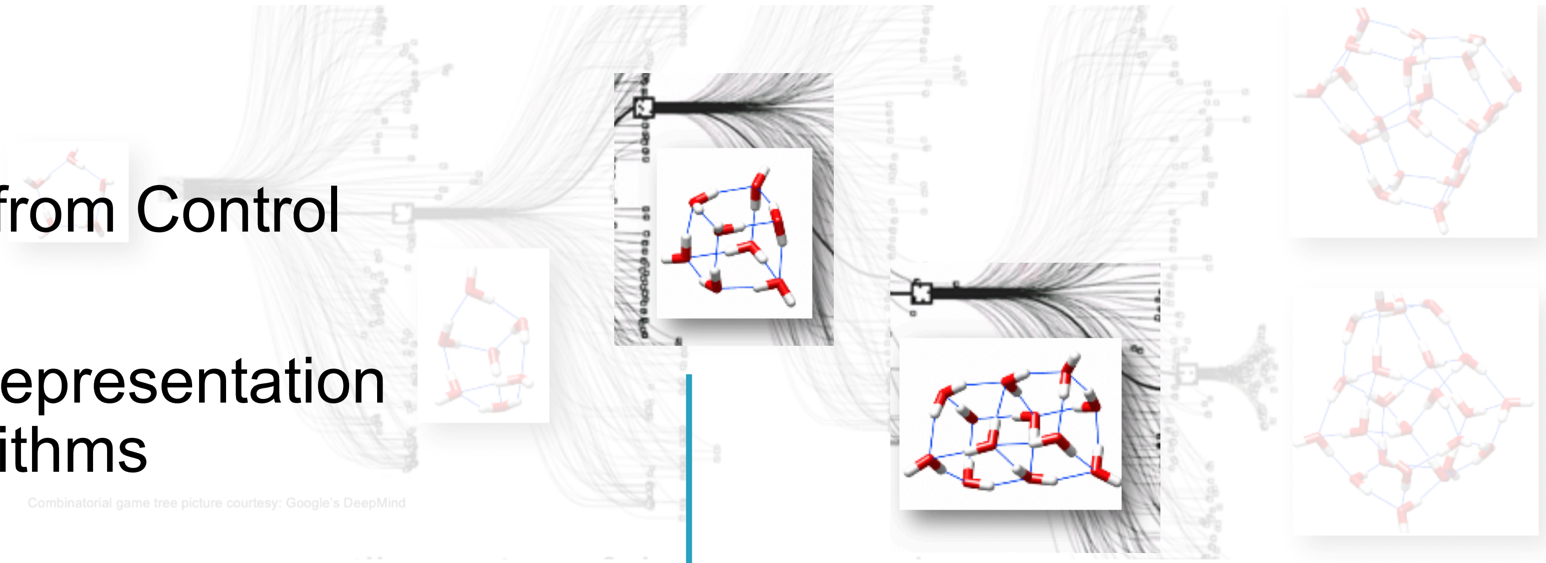
Learning M_{RL} is the focus on this tutorial



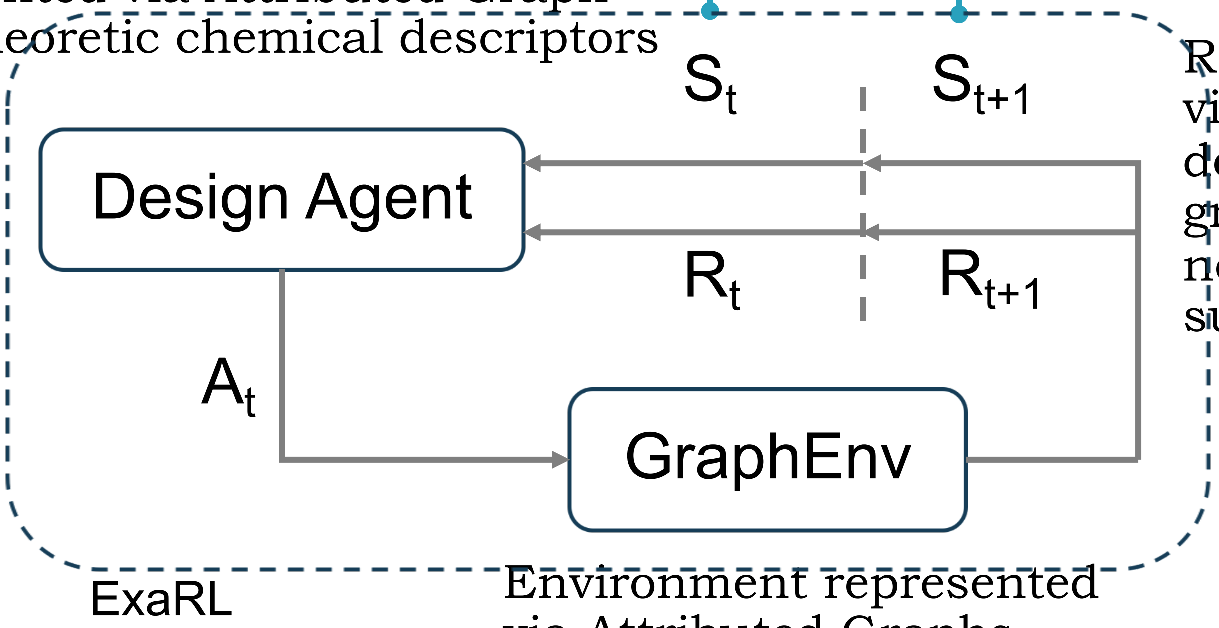
Combinatorial game tree picture courtesy: Google's DeepMind

Learning M_{RL}

- Recollect key RL concepts from Control presentation
- Extends with graph-based representation learning methods and algorithms


Combinatorial game tree picture courtesy: Google's DeepMind

State represented via Attributed Graph and graph-theoretic chemical descriptors

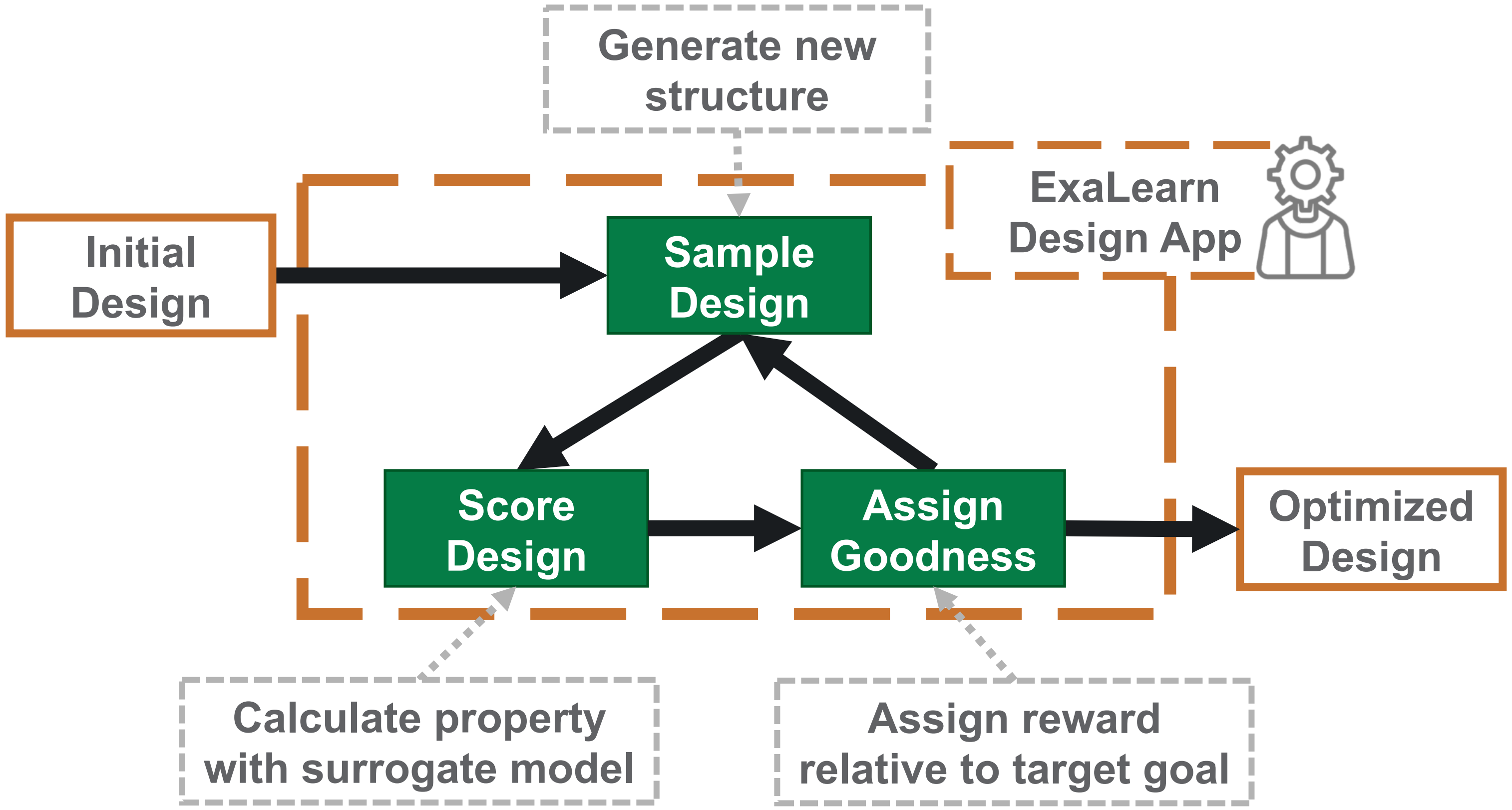


Reward estimated via chemical descriptors and/or graph neural network based surrogate models

Environment represented via Attributed Graphs

Introducing the ML Workflow to Computational Scientists

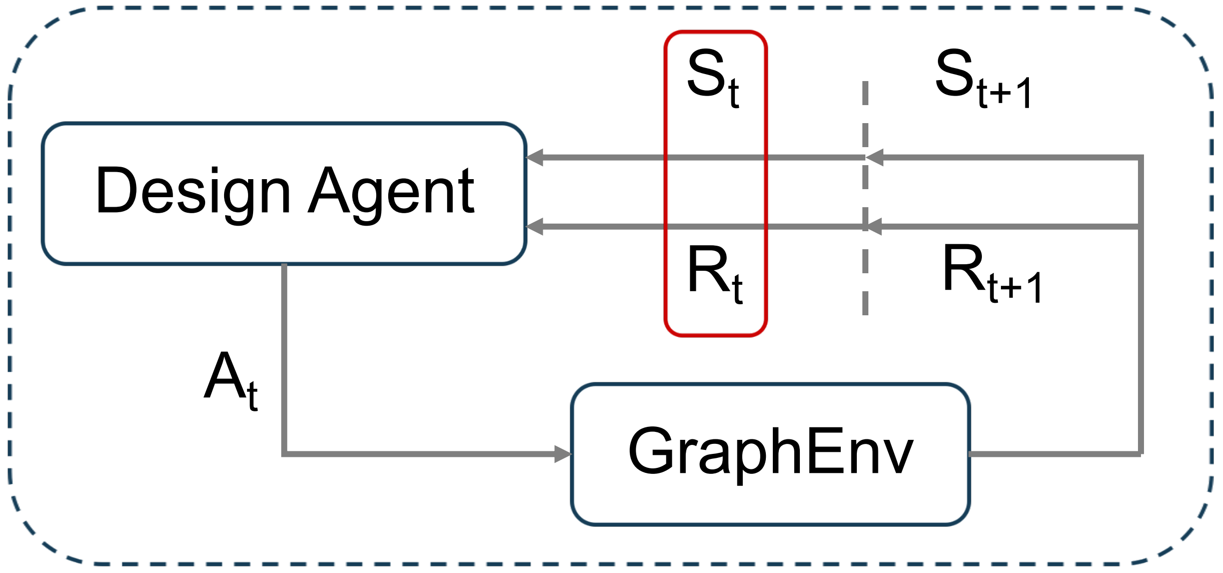
1. Describe the graph-based data representation
2. Show how to train a surrogate model for graph structured data
3. Show how to design graph based chemical descriptors to encode the state and steer rewards

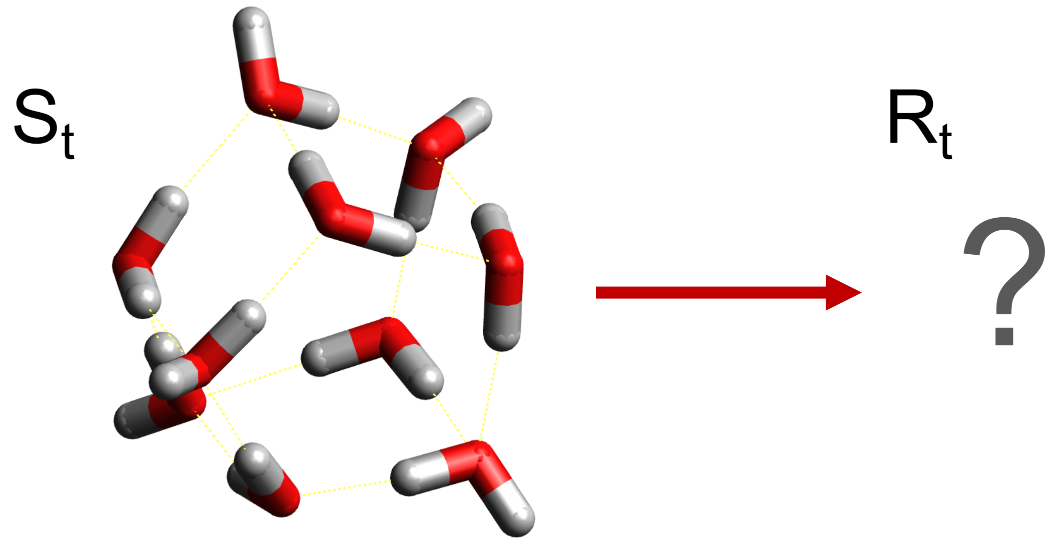


Developing Reward Functions via Graph-Theoretic Chemical Descriptors

Designing domain-aware reward functions

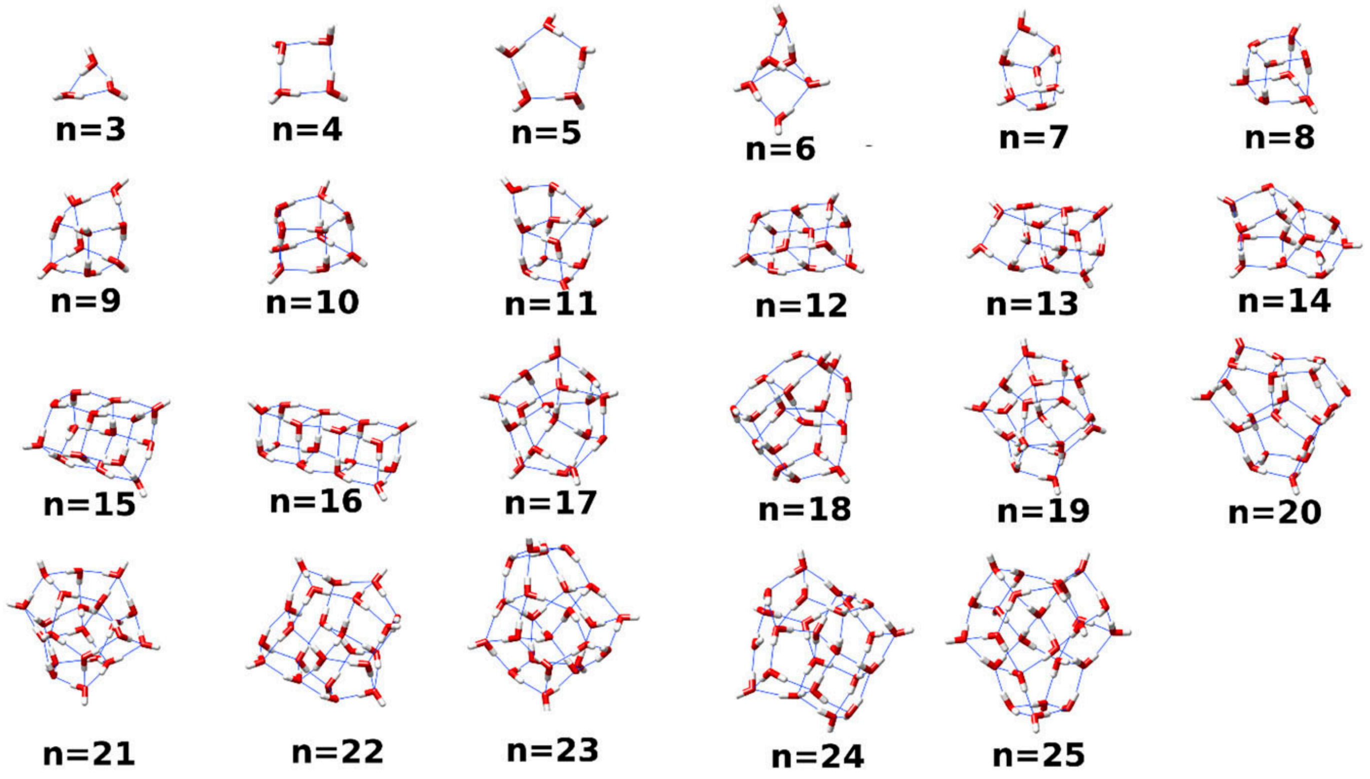
How do you go from state S_t (graph) to reward R_t (scalar)?





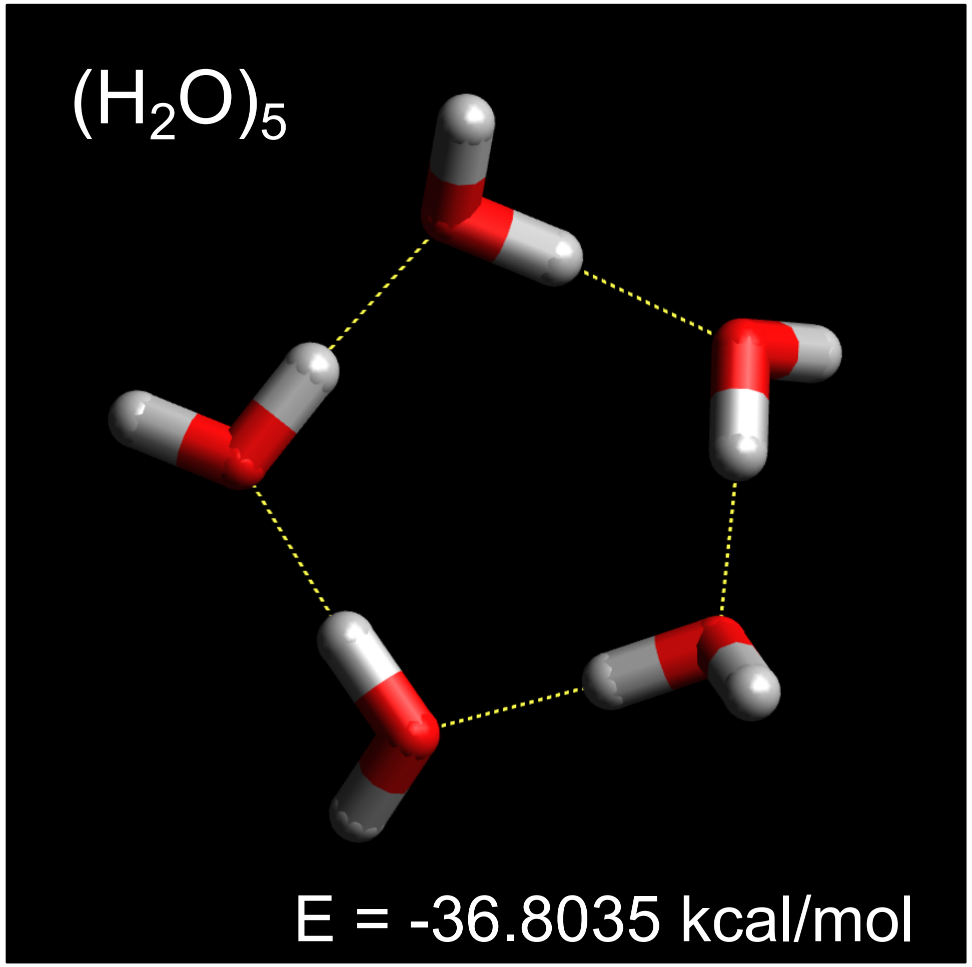
Test Case: Low-lying structures of water clusters

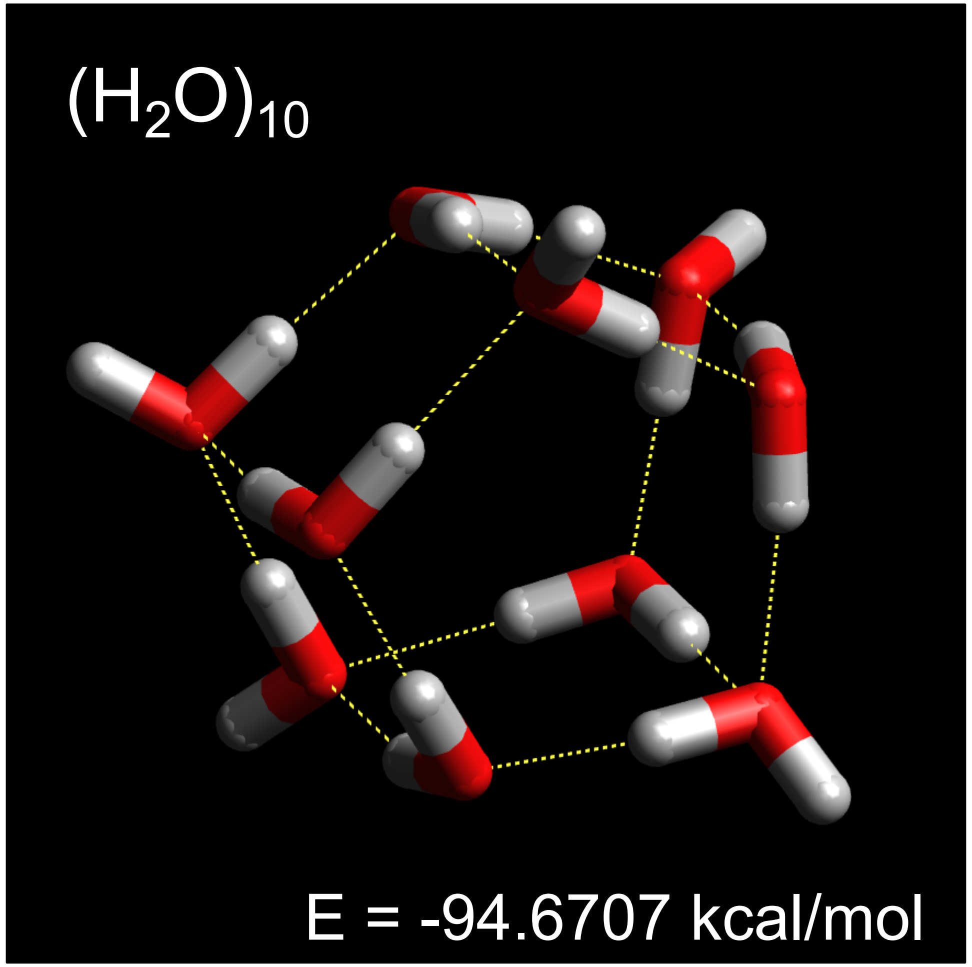
- Test database contains ~5M water clusters with 3–30 molecules, all lying within 5 kcal/mol of the putative minimum for each cluster size
- Database generated through Monte Carlo-type potential energy search
  - Many repeated calculations
  - Many local minima missed



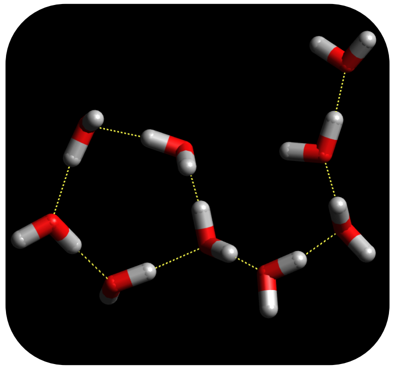
A. Rakshit, P. Bandyapadhyay, J.P. Heindel, S.S. Xantheas. Atlas of putative minima and low-lying energy networks of water clusters *n* = 3 – 25. *J. Chem. Phys.* **151**, 214307 (2019).

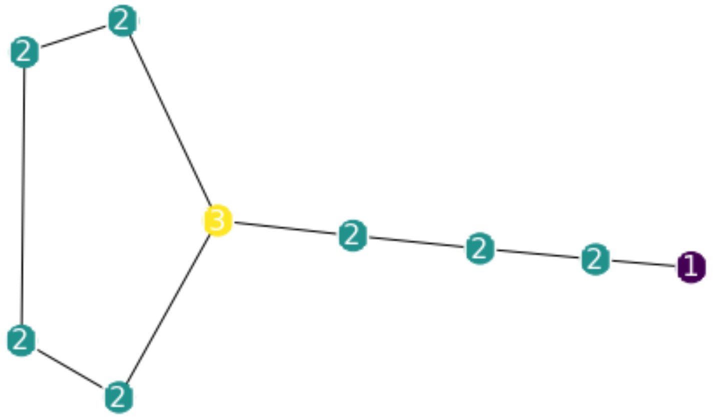
Test Case: $(H_2O)_5 \rightarrow (H_2O)_{10}$

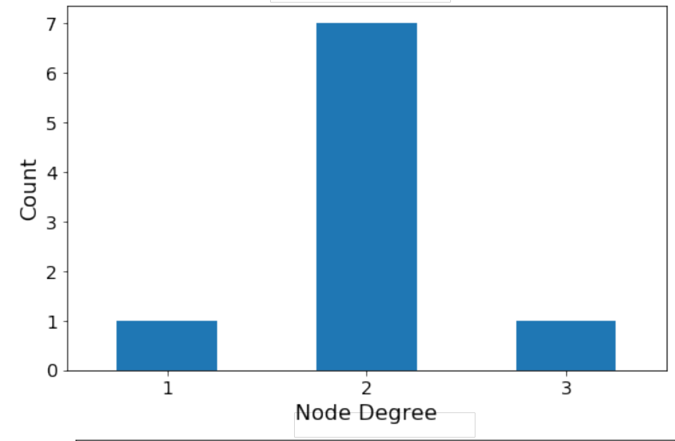


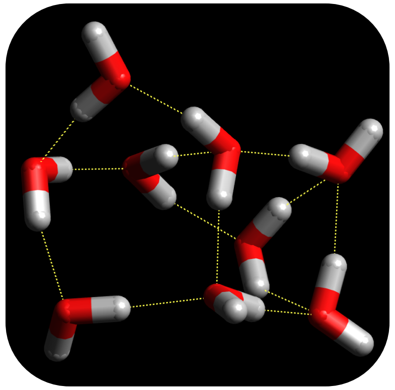


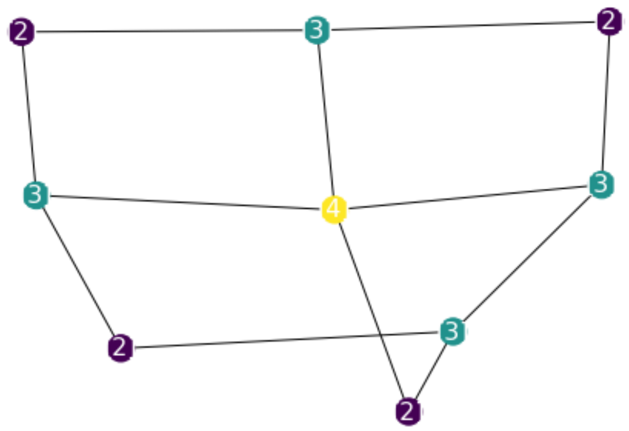
Formalizing the Structure by Degrees

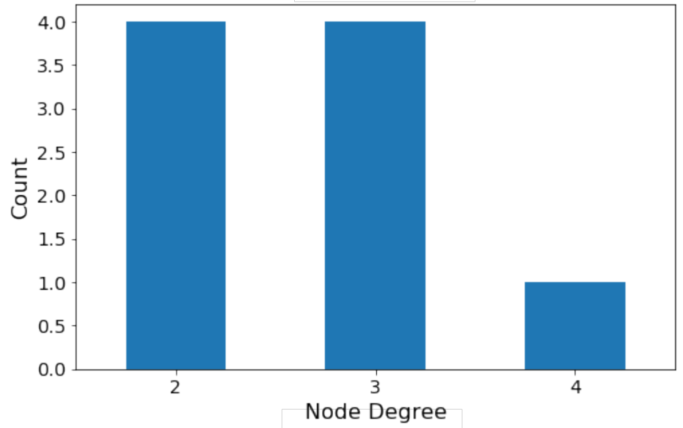


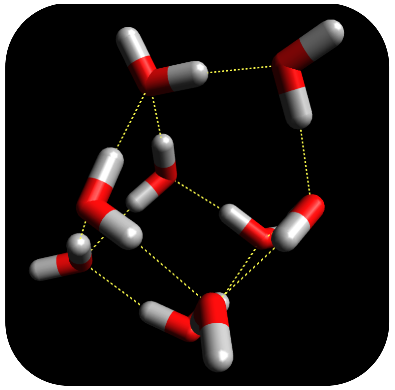


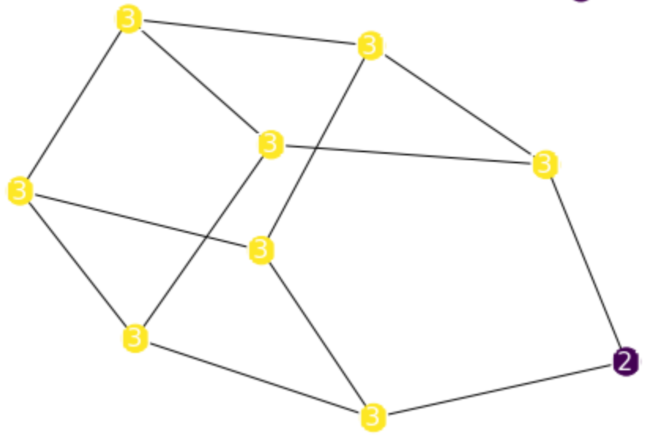


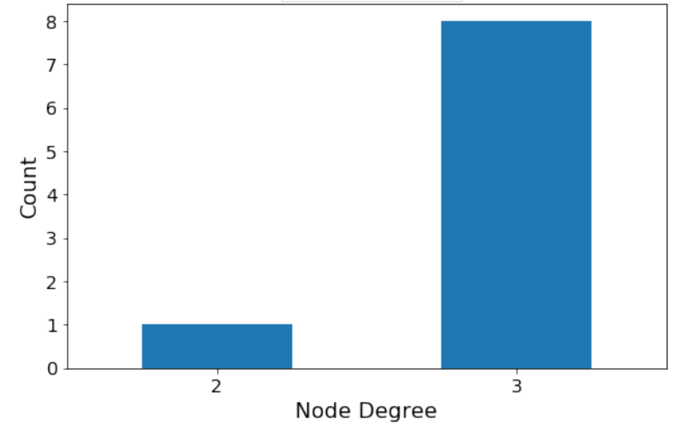


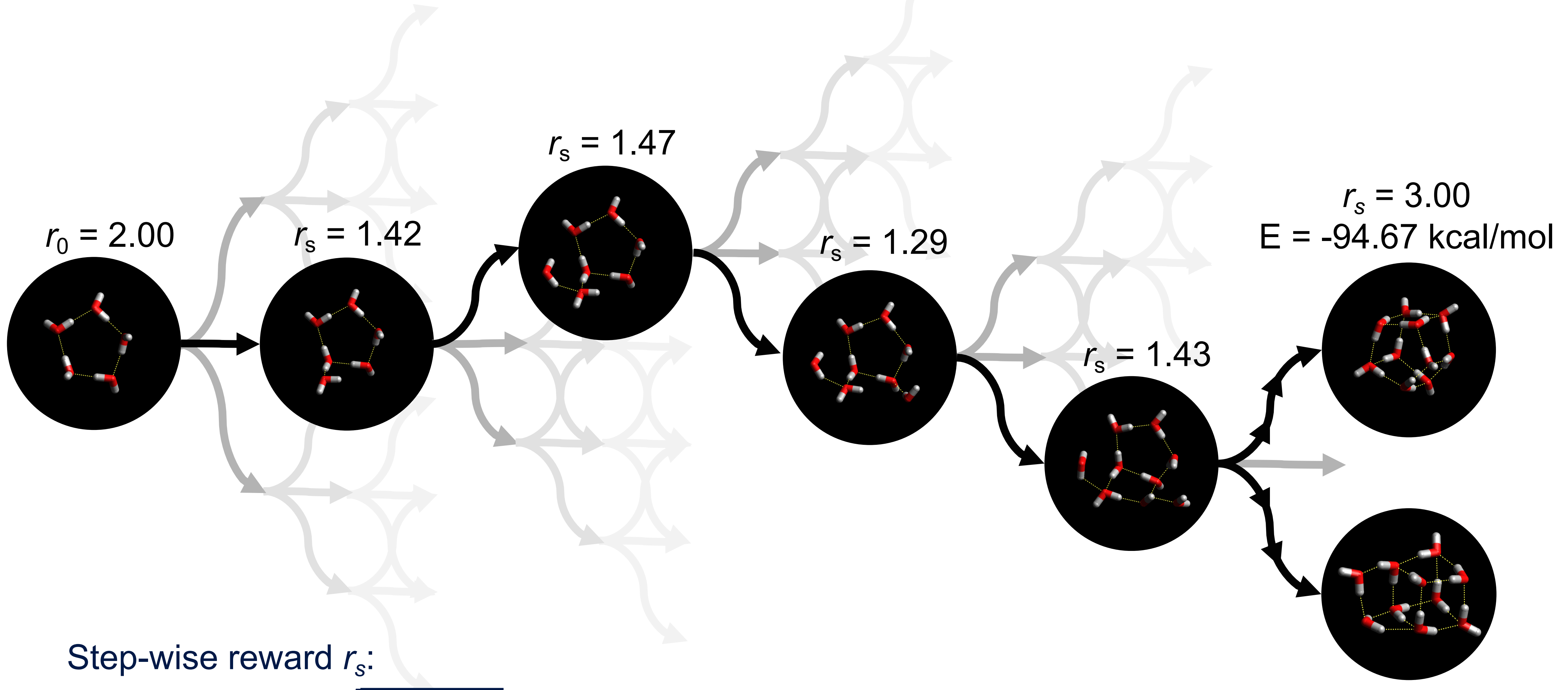








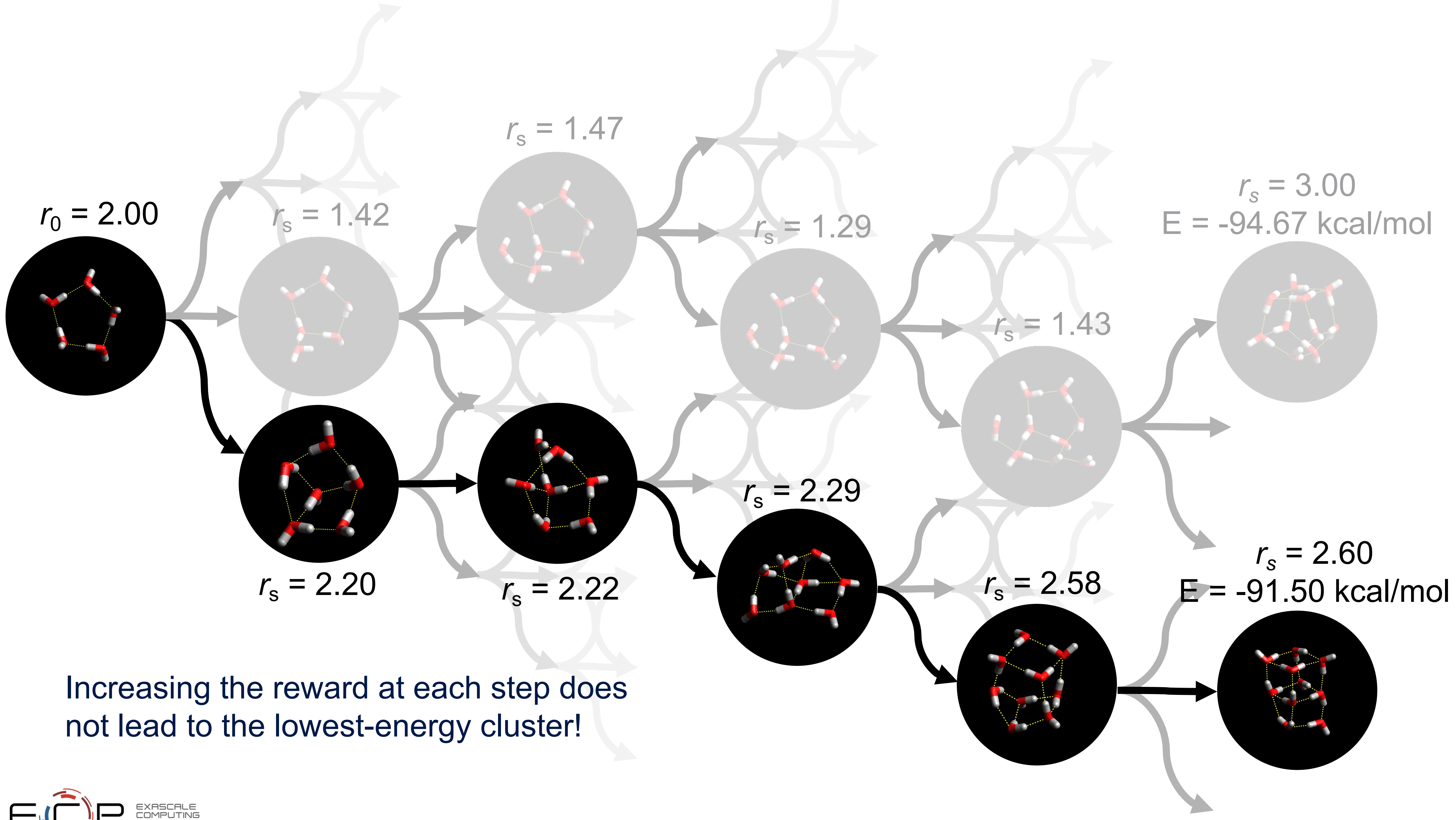




Step-wise reward r_s:

$$r_s = E(D_s) - \sqrt{var(D_s)}$$

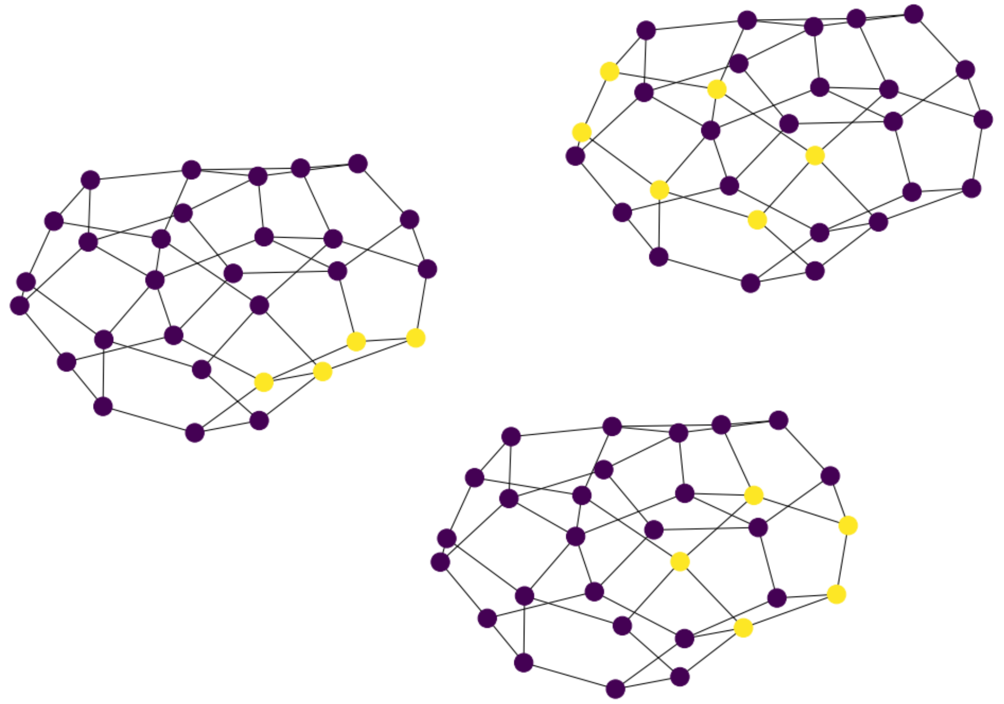
where D_s is the degree distribution at step *s*. As shown for the final steps, the highest reward corresponds to the lowest energy structure.

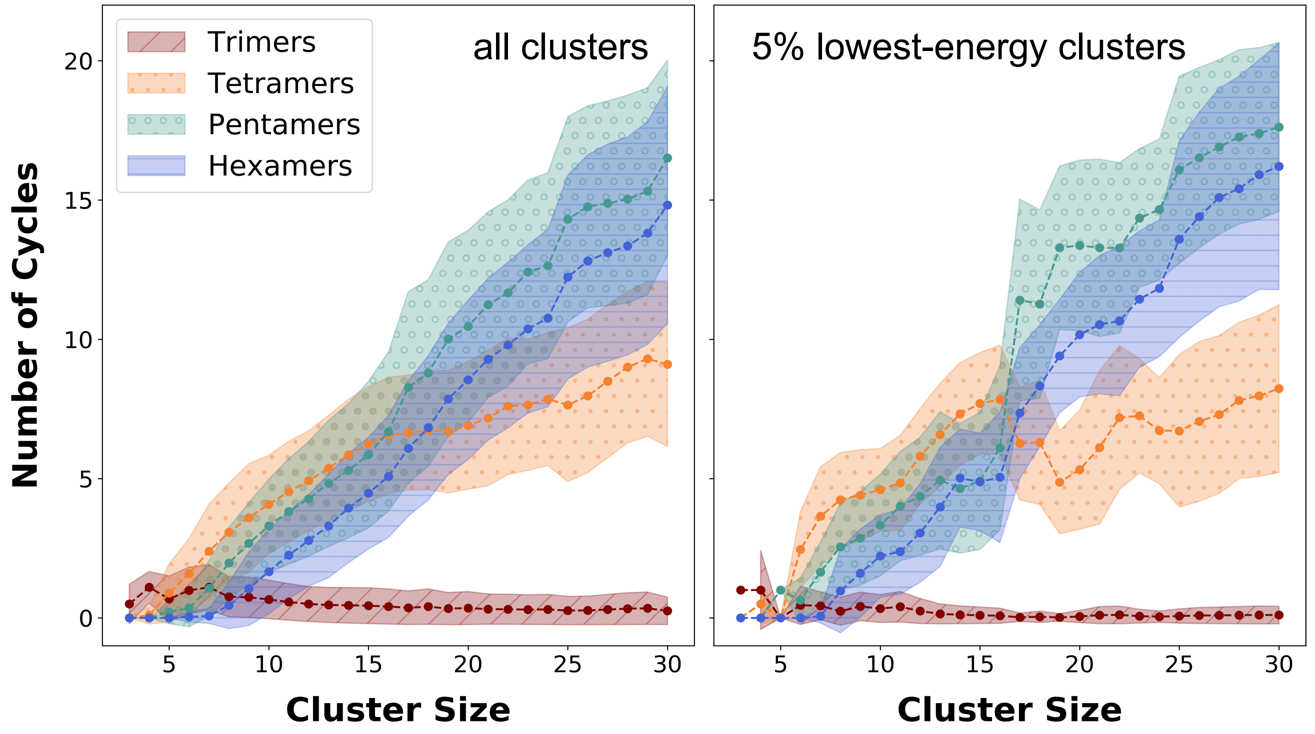


Increasing the reward at each step does not lead to the lowest-energy cluster!

Tuning the Reward though Additional Metrics

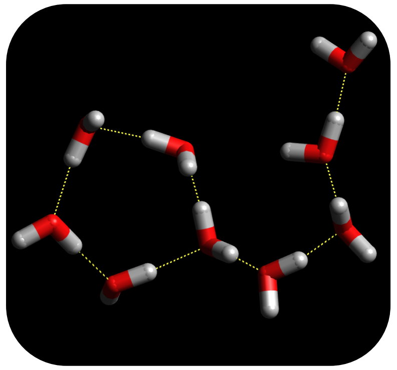
- The reward can be tuned to produce specific structures
- Cluster properties are non-linear as the number of water molecules increases

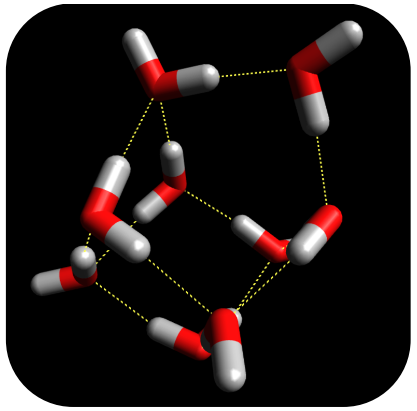




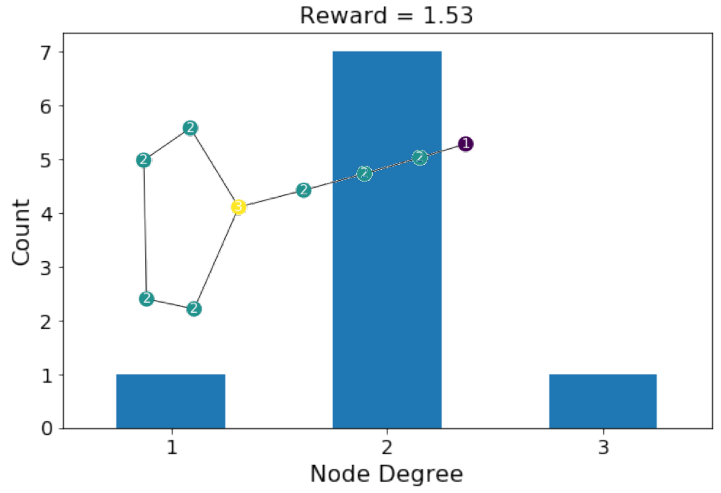
Graph-theoretic Reward Components

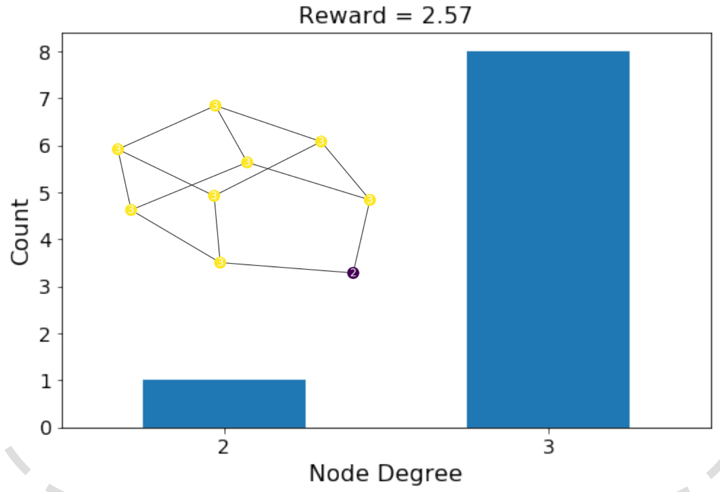
$(H_2O)_9$





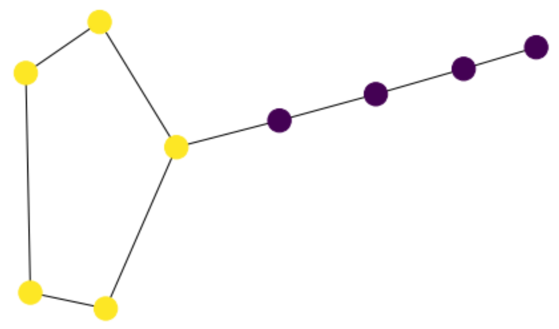
Degree



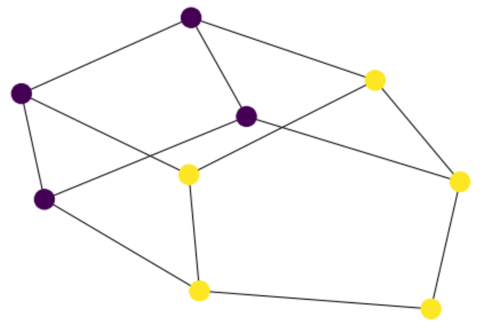


Geometric Cycles

0 Trimers 0 Tetramers
1 Pentamers 0 Hexamers

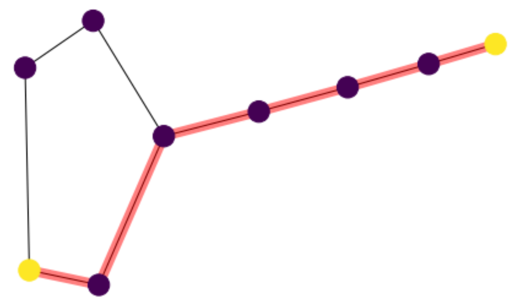


0 Trimers 4 Tetramers
2 Pentamers 3 Hexamers

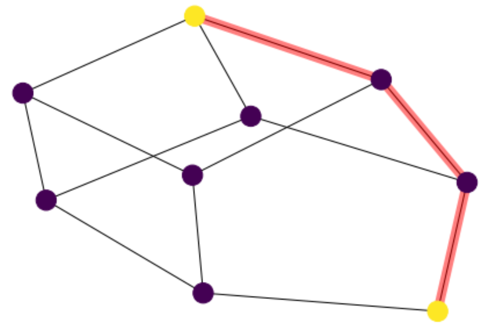


Path Measures

Diameter = 6
Avg Shortest Path Length = 2.75



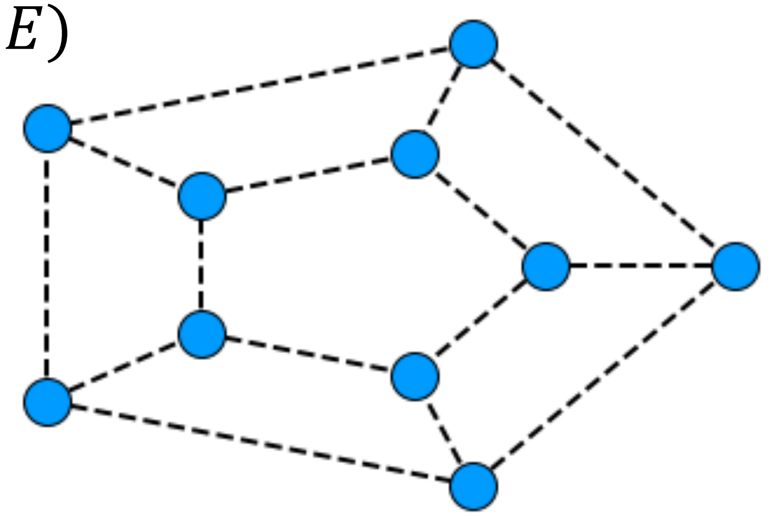
Diameter = 3
Avg Shortest Path Length = 1.89



Introducing Punishments into the Reward

- Bonding Measures
  - Water molecules not connected to the cluster
    - $-|\{v \in V : D(v) = 0\}|$
  - Greater than 4 hydrogen bonds per water molecule
    - $\sum_{v \in V} -relu(D(v) - 4)$

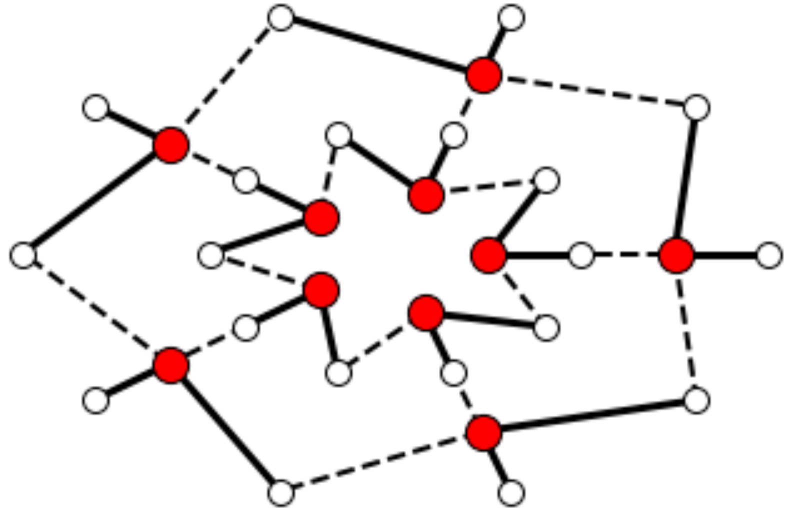
$G = (V, E)$



- Molecular Measures
  - Oxygen-oxygen and hydrogen-hydrogen bonds
  - Incorrect structure for each molecule
    - Each O has exactly 2 covalent bonds
    - Each H has exactly 1 covalent bond
  - Incorrect hydrogen bonding structure
    - Each O can have max 2 hydrogen bonds
    - Each H can have max 1 hydrogen bonds

$$\sum_{v \in V} -(D(v) - x)^2$$

$$\sum_{v \in V} -relu(D(v) - x)$$



EXASCALE COMPUTING PROJECT

Training Graph Surrogate Models at Extreme Scale

A supervised learning model for Molecular Property Prediction

Our tutorial will cover:

- Components of the ML model
- Theory behind our model
- Implementation in HPC
- Optimizing Performance on HPC

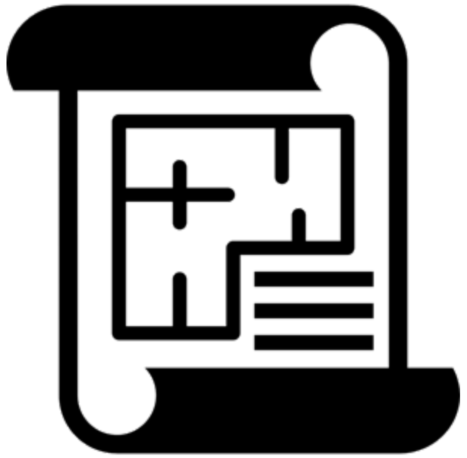
But first, installing the tools on your laptop…

https://github.com/exalearn/design-tutorial

Key Components of a Supervised Learning Model

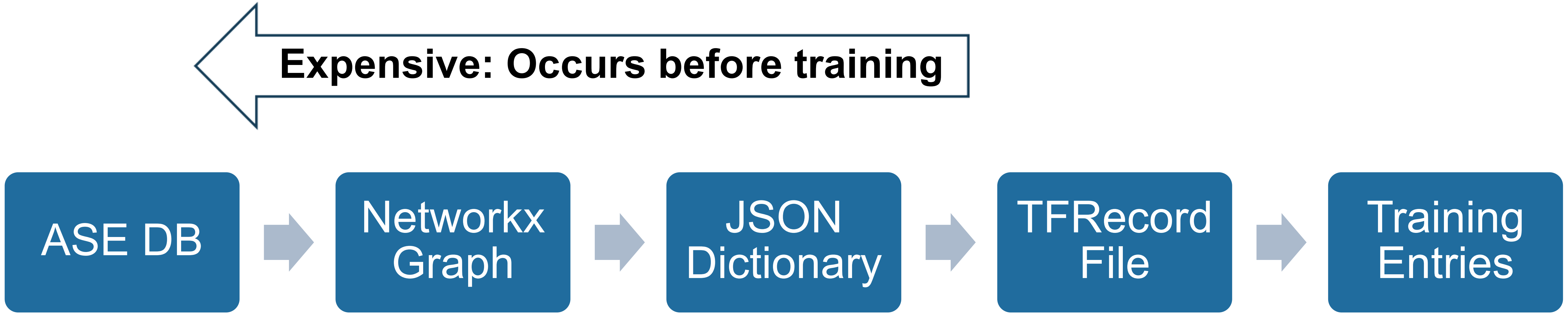
Data Loader

Model Architecture



Execution Engine

Data Ingest and Transformations



1. Review the “0_parse-data.ipynb” notebook
2. Open the `mpnn` directory
3. Review the first part of “0_create-model.ipynb”

Demonstration: Building TFRecord Datasets and Data Loaders

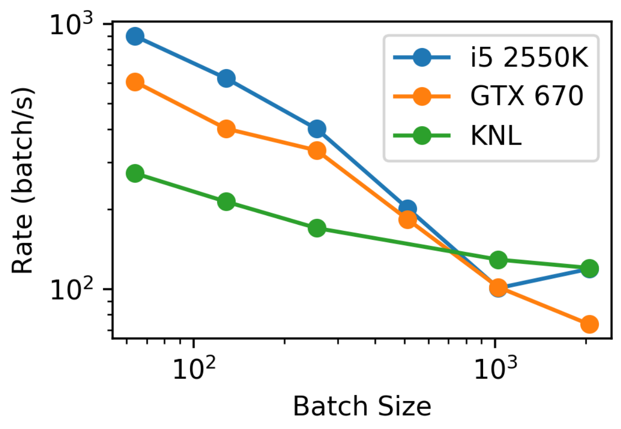
Key concept: How to maximize “batch/second”

Storing as Protobuf-format data

```python
def make_tfrecord(atoms):
    """Make and serialize a TFRecord for in NFP format

    Args:
        atoms (ase.Atoms): Atoms object of the water cluster
    Returns:
        (bytes) Water cluster as a serialized string
    """

    # Make the network data
    features = make_nfp_network(atoms)

    # Convert the data to TF features
    features = dict((k, _numpy_to_tf_feature(v)) for k, v in features.items())

    example_proto = tf.train.Example(features=tf.train.Features(feature=features))
    return example_proto.SerializeToString()
```

Parallel Data Processing

```python
rates = []
para = args.parallel
for b in tqdm(args.batch_sizes, desc=f'Parallel Fixed {para}'):
    r = tf.data.TFRecordDataset(_uncompressed_path).batch(b).map(parse_records, para)\
        .map(prepare_for_batching, para). \
        map(combine_graphs, para).map(make_training_tuple, para).prefetch(8)
    rates.append(test_data_loader(r))
with open(os.path.join(out_dir, f'parallel-fixed-{para}.json'), 'w') as fp:
    json.dump({
        'description': f'Parallel with threads fixed at {para}, prefetching, data on disk',
        'batch_sizes': args.batch_sizes,
        'rates': rates
    }, fp, indent=2)
```



Network Architecture: Message Passing Neural Networks

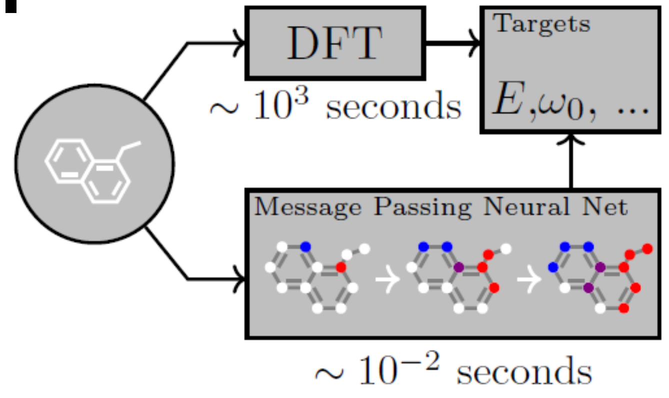


Figure 1. A Message Passing Neural Network predicts quantum properties of an organic molecule by modeling a computationally expensive DFT calculation.

A generalized form of neural networks for graph data, introduced by Gilmer et al. (Google)

[Mention the assumptions]

Existing strategies mostly variants of

$$m_v^{t+1} = \sum_{w \in N(v)} \boldsymbol{M_t}(h_v^t, h_w^t, e_{vw})$$

1. **Gather** messages from neighboring nodes

$$h_v^{t+1} = \boldsymbol{U_t}(h_v^t, m_v^{t+1})$$

2. **Update** node state given messages

$$\hat{y} = \boldsymbol{R}(\{h_v^T | v \in G\})$$

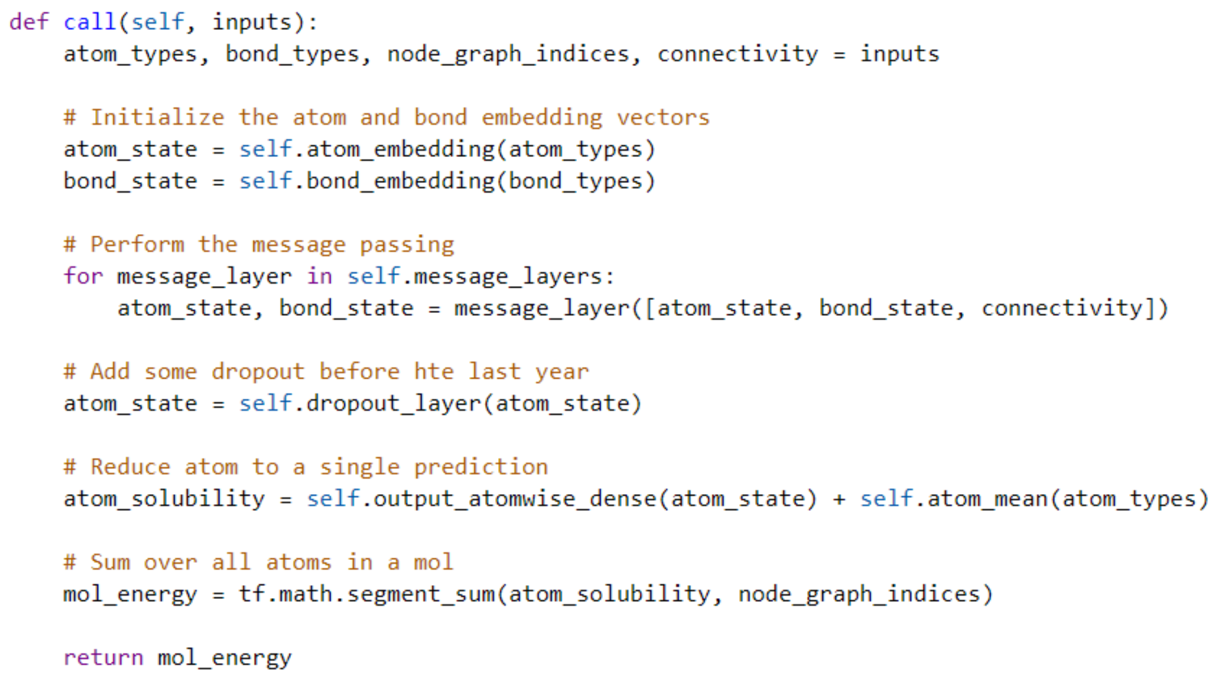
3. **Readout** graph properties given node states

ECP EXASCALE COMPUTING PROJECT

Ref: Gilmer et al. (2017). arXiv:1704.01212v2

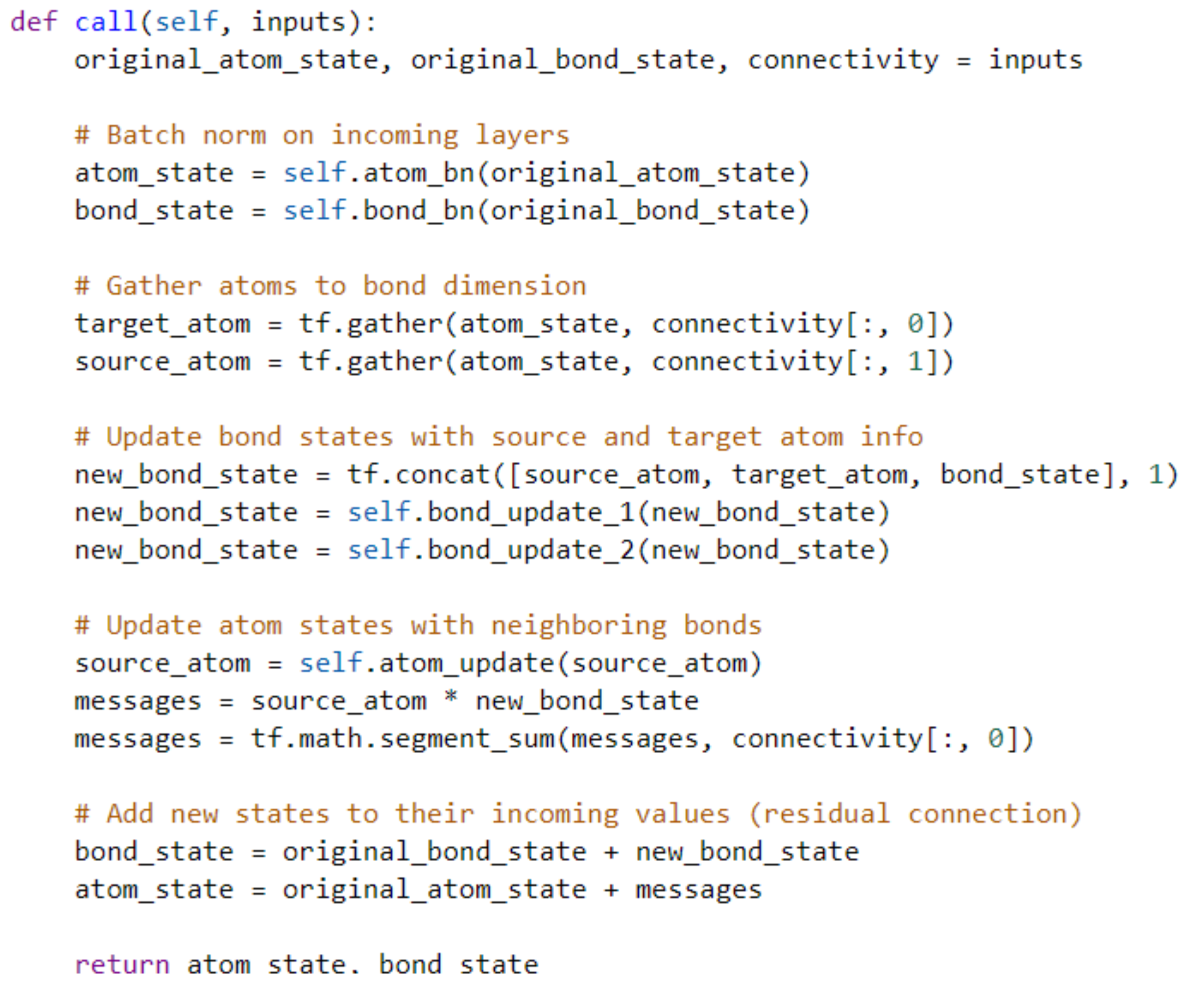
Implementation: Key Bits

Define Network Structure in [tf.]Keras



```python
def call(self, inputs):
    atom_types, bond_types, node_graph_indices, connectivity = inputs

    # Initialize the atom and bond embedding vectors
    atom_state = self.atom_embedding(atom_types)
    bond_state = self.bond_embedding(bond_types)

    # Perform the message passing
    for message_layer in self.message_layers:
        atom_state, bond_state = message_layer([atom_state, bond_state, connectivity])

    # Add some dropout before hte last year
    atom_state = self.dropout_layer(atom_state)

    # Reduce atom to a single prediction
    atom_solubility = self.output_atomwise_dense(atom_state) + self.atom_mean(atom_types)

    # Sum over all atoms in a mol
    mol_energy = tf.math.segment_sum(atom_solubility, node_graph_indices)

    return mol_energy
```

Message Passing as Tensorflow Operations



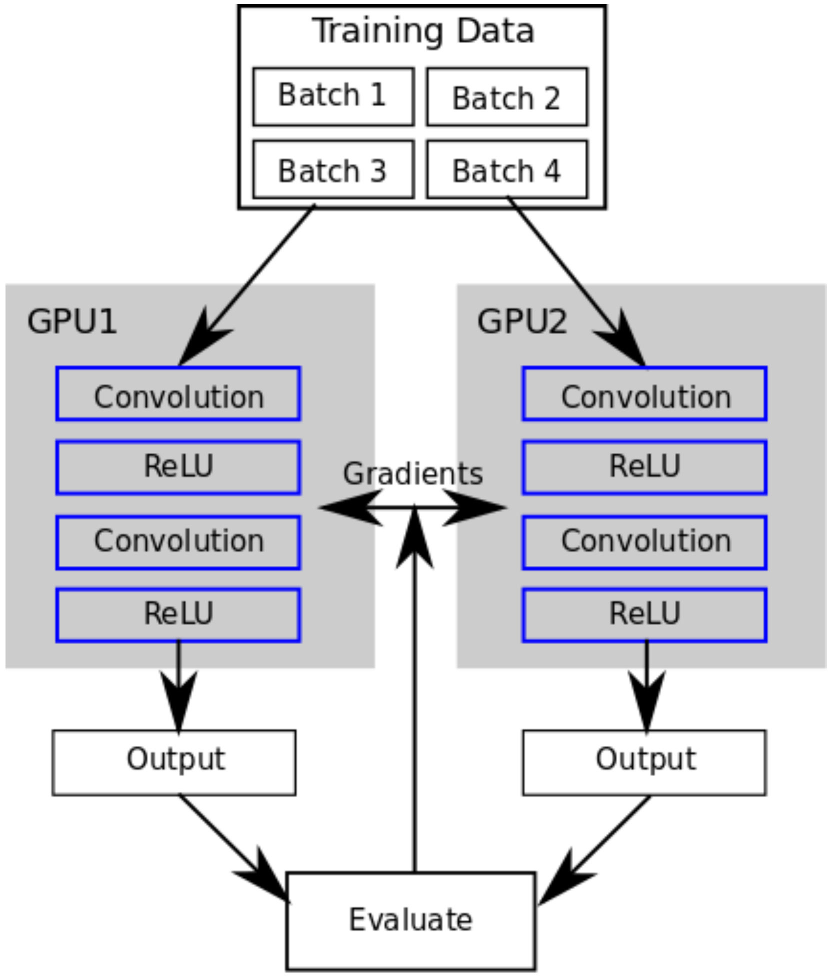
```python
def call(self, inputs):
    original_atom_state, original_bond_state, connectivity = inputs

    # Batch norm on incoming layers
    atom_state = self.atom_bn(original_atom_state)
    bond_state = self.bond_bn(original_bond_state)

    # Gather atoms to bond dimension
    target_atom = tf.gather(atom_state, connectivity[:, 0])
    source_atom = tf.gather(atom_state, connectivity[:, 1])

    # Update bond states with source and target atom info
    new_bond_state = tf.concat([source_atom, target_atom, bond_state], 1)
    new_bond_state = self.bond_update_1(new_bond_state)
    new_bond_state = self.bond_update_2(new_bond_state)

    # Update atom states with neighboring bonds
    source_atom = self.atom_update(source_atom)
    messages = source_atom * new_bond_state
    messages = tf.math.segment_sum(messages, connectivity[:, 0])

    # Add new states to their incoming values (residual connection)
    bond_state = original_bond_state + new_bond_state
    atom_state = original_atom_state + messages

    return atom_state, bond_state
```

EXASCALE COMPUTING PROJECT

Execution Engine: Tensorflow (`tf.keras`) with Horovod

Key concepts:

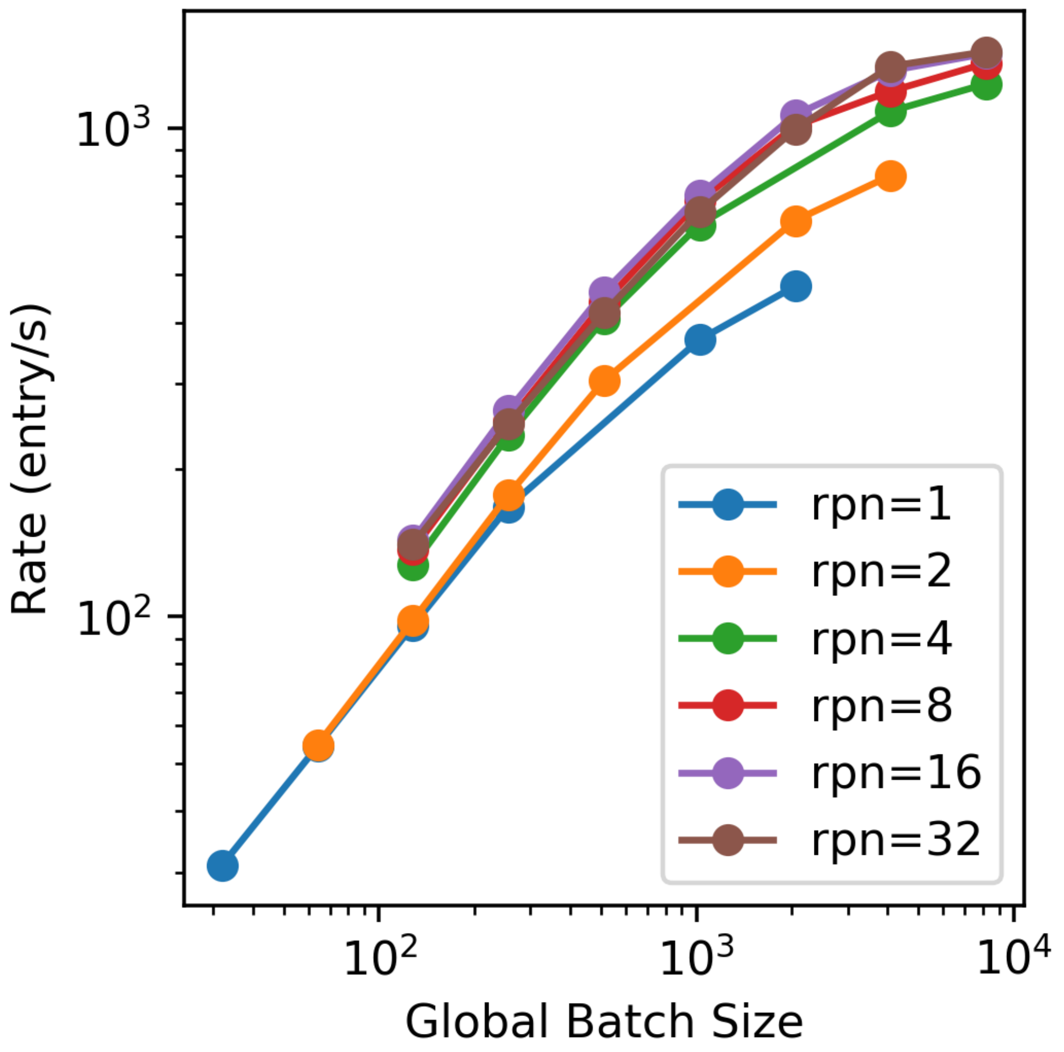
1. Data Parallel Training
   - Each rank has identical weights
   - Allreduce gradients each batch
2. Ring Reduce
   - How Horovod performs the Allreduce
3. Large Batch Size
   - Each node needs a large batch
   - You also need large learning rates



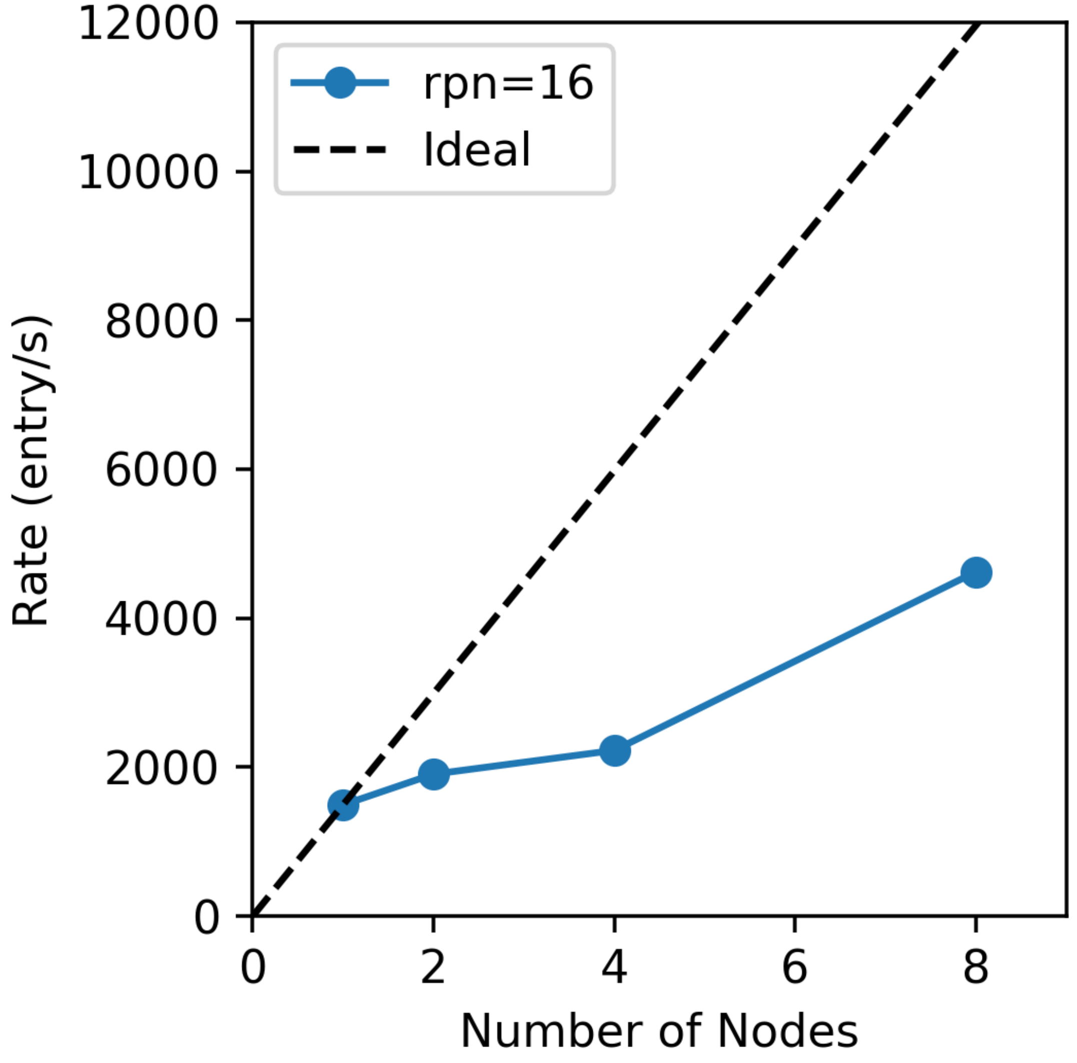
Reference: https://resources.rescale.com/deep-learning-with-multiple-gpus-on-rescale-torch/

Optimizing Training Performance

Tuning Batch Size and Intra-Node Replicas



Optimizing Internode Parallelism



Summary: Optimization Tips and Tricks

MPNNs for Water Cluster Energy

What data?

4.4 M networks, stored as TFRecords

What model?

Tensorflow gather/reduce operations

Trained how?

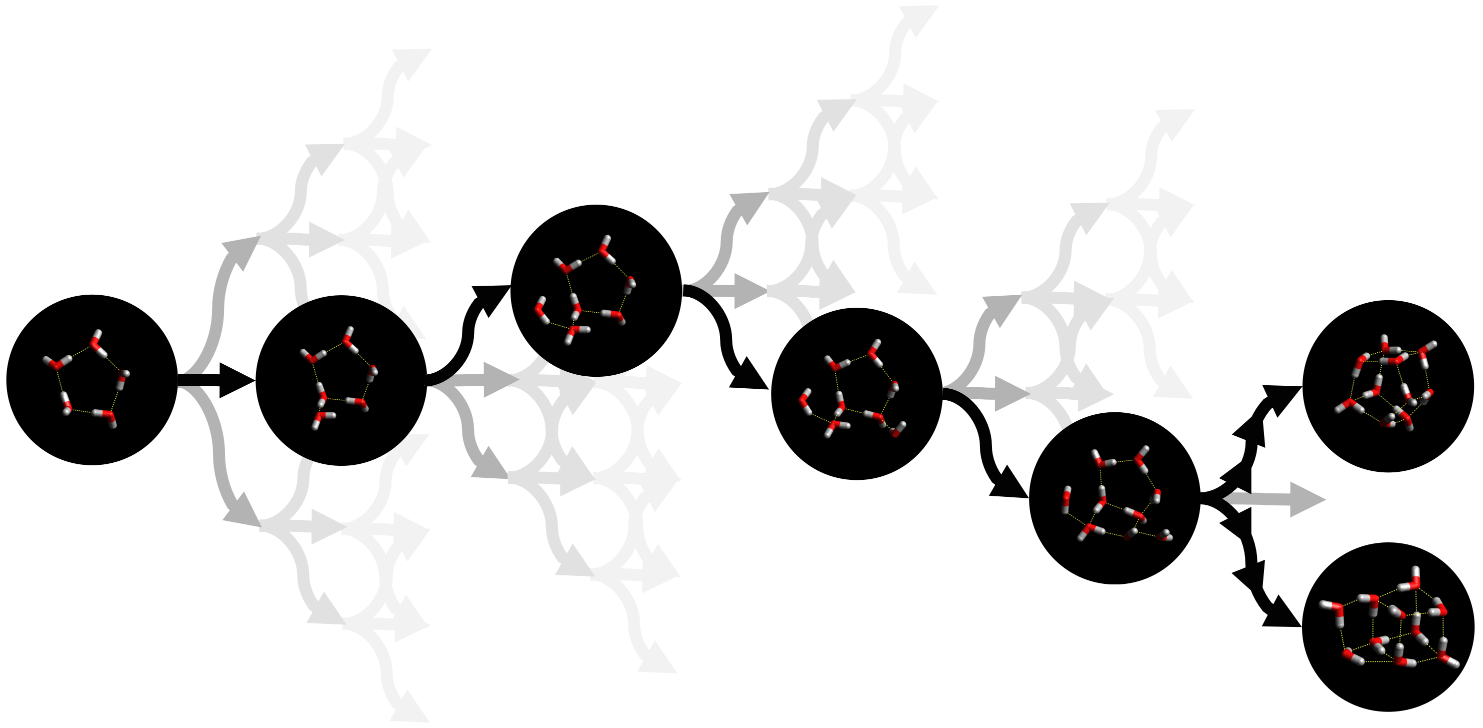
Horovod on ALCF’s Theta

Training it Quickly on Cray XC40

- Batch sizes of ~1024 for optimal parallelism
- Parallel data loader mandatory for manycore architectures
- Exploit intranode parallelism for models too small for KNLs

Summary

- Generic Design framework aimed at multiple chemistry applications
- Developing domain-aware RL models via graph-theoretic rewards
- Scaling of end-to-end framework development under progress
- Ready to start integration with application partners



Thank you!

EXASCALE COMPUTING PROJECT

National Nuclear Security Administration

U.S. DEPARTMENT OF ENERGY | Office of Science