JCTC Journal of Chemical Theory and Computation

pubs.acs.org/JCTC Article

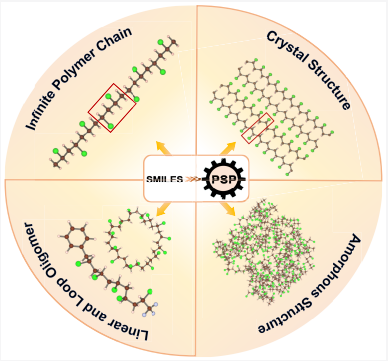
Polymer Structure Predictor (PSP): A Python Toolkit for Predicting Atomic-Level Structural Models for a Range of Polymer Geometries

Harikrishna Sahu, Kuan-Hsuan Shen, Joseph H. Montoya, Huan Tran,* and Rampi Ramprasad*

Cite This: https://doi.org/10.1021/acs.jctc.2c00022 Read Online

ACCESS | Metrics & More | Article Recommendations

ABSTRACT: Three-dimensional atomic-level models of polymers are the starting points for physics-based simulation studies. A capability to generate reasonable initial structural models is highly desired for this purpose. We have developed a python toolkit, namely, polymer structure predictor (PSP), to generate a hierarchy of polymer models, ranging from oligomers to infinite chains to crystals to amorphous models, using a simplified molecular-input line-entry system (SMILES) string of the polymer repeat unit as the primary input. This toolkit allows users to tune several parameters to manage the quality and scale of models and computational cost. The output structures and accompanying force field (GAFF2/OPLS-AA) parameter files can be used for downstream *ab initio* and molecular dynamics simulations. The PSP package includes a Colab notebook where users can go through several examples, building their own models, visualizing them, and downloading them for later use. The PSP toolkit, being a first of its kind, will facilitate automation in polymer property prediction and design.



1. MOTIVATION AND SIGNIFICANCE

During the last decades, physics-based simulation methods have risen in prominence in polymer science and engineering.[1−5] Ranging from electronic-structure-based computations that involve atomic-scale details to coarse-grained molecular dynamics (MD) simulations where mesoscopic-scale dynamical processes may be probed, these methods need atomic-level polymer models as a required input. This very first step is a major bottleneck in polymer simulations. On the one hand, polymers are largely amorphous, and suitable structural models to start a simulation are generally not available. On the other hand, even if we assume that simple linear polymers, e.g., polyethylene (PE) and polyvinyl chloride (PVC), can be modeled as crystals, predicting their crystal structures and chain-level conformations are much more challenging than in hard materials.[6]

In practice, polymer models must be handcrafted with a great deal of human intervention and intuition. For example, starting from the atomic connectivity information on a polymer, the atomic structure of its repeating unit may be constructed using cheminformatics software such as RDKit[7] and Open Babel.[8] Next, other toolkits like STK[9] are needed to simply (and perhaps also randomly) replicate the atomic structure of a polymer repeating unit in order to create (finite) polymer chains, i.e., oligomers. Large three-dimensional (3D) models of this polymer can then be created by randomly placing a number of these chains in a simulation box, reproducing the desired density.[10−12] In addition, a system-specific force field (FF) parameter file, which includes information about atom types, bond connectivity, masses, and so on, is also required for MD simulations. Creating such a file for large models is nontrivial. There are a few existing toolkits for this purpose,[13,14] but their capabilities are limited to a single chain or necessitate a configuration file, preventing true automation. Notably, the polymer builder module of CHARMM-GUI[15,16] allows generating models for finite polymer chains, solutions, and melt systems with necessary topology, force field parameters, and input files for downstream MD simulations. However, its capabilities are restricted to a specific set of monomers, end-cap groups, and solvents, limiting its application to a specific chemical space. Apart from this, these large models cannot be utilized to calculate their electronic structures using high-level *ab initio* methods, which requires much smaller and more precise atomic models. Presently, there is no practical tool that can accept the atomic connectivity information on the repeating unit, usually encoded in terms of a simplified molecular-input line-entry system (SMILES),[17] to create user-controlled small- or large-scale polymer models for any desired systems that are suitable for multiple computational objectives and schemes.

In this paper, we release the first version of Polymer Structure Predictor (PSP), an open-source python-based toolkit

Received: January 7, 2022

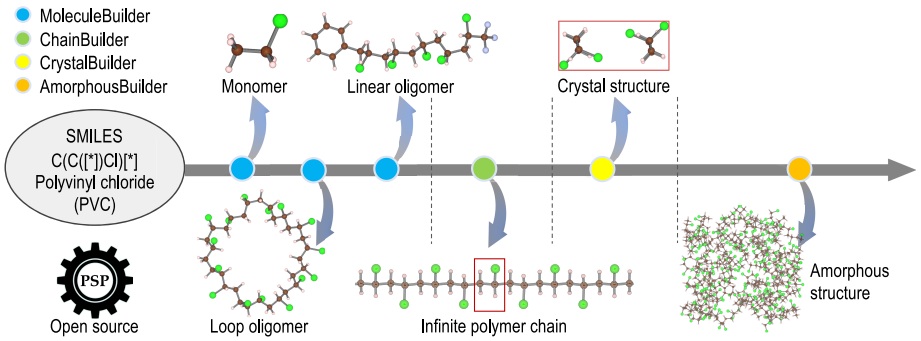


Figure 1. An overview of the PSP workflow. Starting from a polymer SMILES, `MoleculeBuilder` and `ChainBuilder` construct oligomers/polymer chains and then `CrystalBuilder` and `AmorphousBuilder` build candidates of crystal and amorphous models, respectively. The red rectangle and box are the unit cells that will be repeated due to the imposed periodicity. In polymer SMILES, asterisks ([*]) are used to indicate the linking atoms, through which adjacent repeating units are combined covalently. More details on polymer SMILES can be found at www.polymergenome.org.

for predicting a hierarchy of increasingly sophisticated polymer models, e.g., repeating units, finite oligomers, infinite chains, crystals, and amorphous structures. Starting from a polymer repeating unit SMILES, the atomic structure of its repeating unit is predicted and repeated to create on-demand (finite) oligomers. In addition, periodic boundary conditions are also imposed on the repeating unit structure, creating an *infinite* polymer chain model. It should be noted that PSP only creates a unit cell of a chain model, and supercell structures can be constructed using external packages such as VESTA.[18] Next, polymer crystals are created by assembling oligomers or infinite chains in a low-energy (thermodynamically stable) pattern.[6] At the top level of the hierarchy, amorphous structures with a desired density are generated by creating individual linear and loop oligomers and packing them in a specified box using the PACKMOL package.[11]

To ensure the reliability of polymer models, they were compared with known structures, and various properties of polymers, such as bandgap, i.e., the energy difference between the top of the valence band and the bottom of the conduction band, ring-opening polymerization enthalpy, and glass transition temperature, were computed using *ab initio* and MD simulations and verified against reported experimental data. Also, the computational time required for the model generation was benchmarked using a sufficiently large polymer SMILES data set. This version of PSP has been used in a recently developed polymer version[19] of computational autonomy for materials discovery (CAMD),[20] an artificial-intelligence-based platform for discovering polymers with targeted performance.

The main objective of PSP is to provide a practical, complete, and integrated toolkit for creating a pool of (candidates of) polymer models in an automatic, high-throughput, and efficient manner, minimizing human bias and intervention in the process. Notably, the universal force field (UFF)[21] is a widely applicable classical force field in which parameters are estimated using simple rules based on the element, hybridization, and connectivity, capable of reproducing the majority of structural features across the periodic table. For this reason, PSP uses the UFF and optimization algorithms implemented in RDKit[7] for the preliminary geometry optimizations of the models, leaving any heavy calculations, e.g., those at the first-principles level, for users in an on-demand fashion. PSP-generated polymer structures can be utilized for performing *ab initio* quantum mechanical computations using several available packages such as VASP,[22] ORCA,[23] GAMESS,[24] etc. LAMMPS[25] data files, including force fields such as the optimized potentials for liquid simulation-all atom (OPLS-AA)[26] and the general AMBER force field (GAFF2),[27] are also generated that can be directly used for MD simulations. It is worth noting that PSP has already been used to generate databases for a variety of polymer properties that contribute to the Polymer Genome[28] (www.polymergenome.org) and Khazana (https://khazana.gatech.edu). For pedagogical purposes, a Colab notebook is created, providing users a complete set of installation instructions and practical examples. It should be noted that this version of PSP is only applicable to homopolymers. In the future, it will be further developed to include a wide variety of copolymers, polymer networks, and ladder polymers. Given that PSP is the first of its kind, we believe that this toolkit will be useful for the polymer scientist community.

2. PROBLEMS, SOLUTIONS, AND REMAINING CHALLENGES

The PSP package, whose overall workflow is given in Figure 1, was designed to create a hierarchy of polymer models from SMILES strings of the repeating unit (see Table 1). It is worth noting that, for ring monomers, we must open the ring and provide a linear SMILES string. Finite and infinite polymer chain models are generated from a given SMILES string by the `MoleculeBuilder` and `ChainBuilder` modules. In the next step, candidates for polymer crystal and amorphous structures are created by the `CrystalBuilder` and `AmorphousBuilder` modules, respectively. These candidates can then be used for further *ab initio* or MD simulations to compute various structural, electronic, and mechanical properties of small molecules and polymers. Technical details of these modules are described in the next sections.

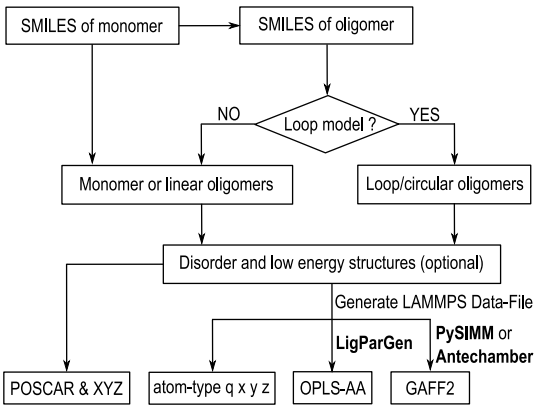
Table 1. PSP Modules and Their Applications

module	polymer models
`MoleculeBuilder`	monomer, linear and loop oligomers
`ChainBuilder`	infinite polymer chain and oligomers
`CrystalBuilder`	crystal structures for small molecules and polymer chains
`AmorphousBuilder`	amorphous structures

https://doi.org/10.1021/acs.jctc.2c00022
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

2.1. Finite Polymer Chain Prediction. Finite polymer chain models are constructed from a polymer SMILES string in which two linking atoms are indicated with asterisks ([*]). The workflow depicted in Scheme 1 is implemented in the

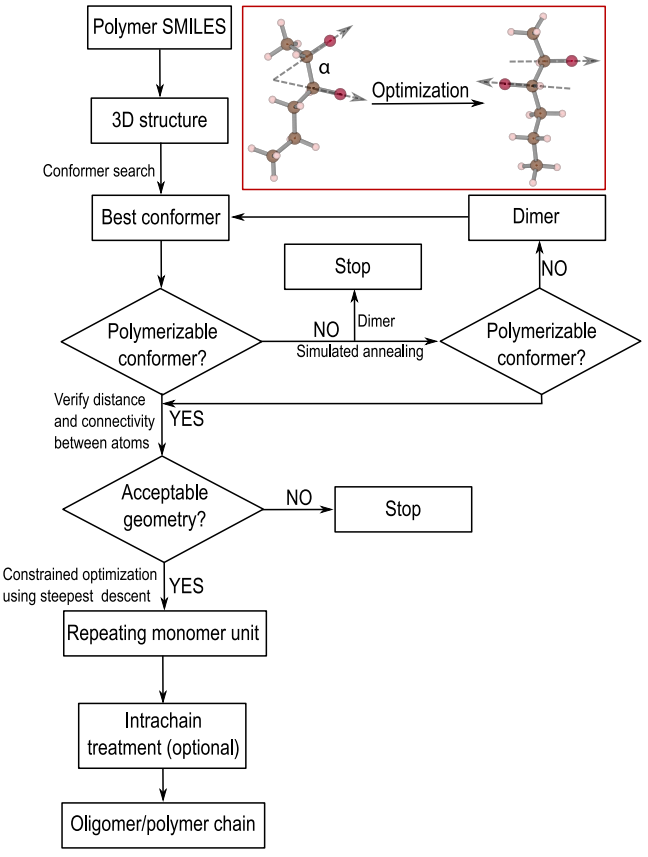
Scheme 1. Schematic Workflow of the `MoleculeBuilder` (MB) Module



`MoleculeBuilder` module of PSP. First, SMILES strings of the desired oligomers are generated by joining SMILES of the repeating unit. If a loop model is desired, the two end linking atoms are connected; otherwise, dummy atoms ([*]) are replaced with hydrogen atoms or end-cap groups. Finally, hydrogen atoms are added, and Cartesian coordinates of linear/loop models are generated, followed by conformation search and geometry minimization using UFF as implemented in RDKit.[7] Due to the limitation of the conformation search method and force field, structures generated by RDKit are sometimes adequately organized but lack essential intrachain non-covalent interaction. To mitigate this issue, PSP optionally performs a quick MD simulation at a constant number of particles, volume, and temperature (NVT) for 15 ps with a time step of 0.5 fs, followed by geometry optimization using the GAFF2. This simulation is carried out with the PySIMM[29,30] and LAMMPS[25] packages. 3D models for multiple conformers of linear or circular oligomers are the output of PSP.

2.2. Infinite Polymer Chain Prediction. Oligomer and polymer chain structures are constructed from the repeating unit predicted from SMILES. Such a polymer building block should be thermodynamically preferable while being suitable for imposing a periodic condition. The workflow depicted in Scheme 2 was designed for meeting these requirements. In the first step, the asterisks [*] are replaced with real (dummy) atoms, i.e., Cl or O when a single or a double bond, respectively, which is needed for linking the blocks. Then, the 3D structure of this "molecule" is generated using RDKit package.[7] Next, this molecule is optimized by rotating the rotatable (single) bonds for minimizing the energy computed from UFF[21] while, at the same time, maximizing α, the angle between two vectors originated from two sets of the dummy and linking atoms (see the inset of Scheme 2). This optimization step is motivated by the observation that single bonds can be rotated quite easily, i.e., by overcoming relatively low energy barriers, for exposing the sinusoidal-like energy profile of organic molecules with comparable minima.[31−34] Simulated annealing (SA)[35] is a stochastic global search optimization algorithm that is particularly useful when there

Scheme 2. Schematic Workflow of the `ChainBuilder` (ChB)[a] Module



[a]The angle (α) between two vectors originated from two sets of the dummy and linking atoms is shown (in red box), which is a crucial factor in simulated annealing.

are a large number of local minima, as there are in this case. For this purpose, we used the SA method to optimize the following objective function

$$E = E_1 + E_2 + E_3 \quad (1)$$

Here, $E_1 = E_{UFF}$ is the energy of the molecule obtained using UFF,[21] while $E_2 = E_{UFF}(1 - \alpha/180)$ is a penalty term that drives α toward 180°. The last term, i.e., E_3, was defined to be zero if the atomic connectivity is correct and $10 \times E_{UFF}$ otherwise, in order to prescribe a high cost for any violation of the connectivity and ensure the correctness of the chemical structure. If this procedure fails to get the desired conformer, a flexible dimer (non-periodic) is built from the original repeating unit and considered as a new molecule, and then all of the previous steps are repeated. The idea is that dimerization could provide more flexibility to the backbone of the molecule by increasing the number of single bonds while preserving the desired atomic connectivity. The obtained molecule is further optimized using the steepest descent method, keeping the positions of the dummy and linking atoms fixed.

Since PSP rotates single bonds randomly, focusing on maximizing α, sometimes it leads to a twisted polymer (often helical) backbone. Although the geometry of an isolated polymer chain may not resemble its conformation in a bulk structure,[6] we want to avoid helical geometries because it may lead to additional intrachain van der Waals interactions.

https://doi.org/10.1021/acs.jctc.2c00022
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

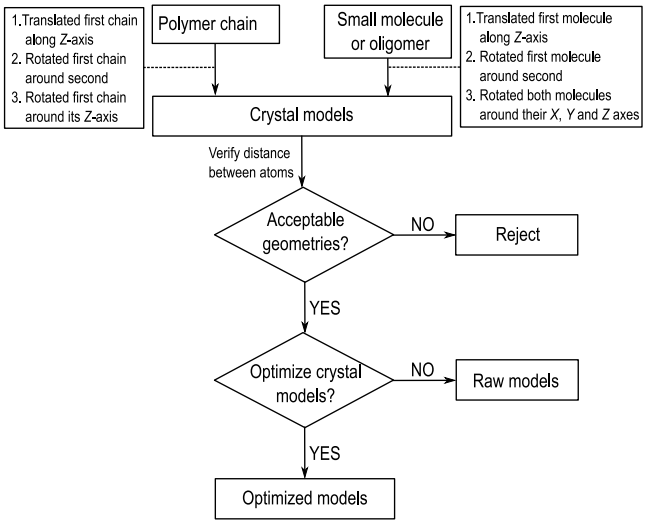
Therefore, PSP attempts to make the backbone of a monomer straight by gradually moving two sets of the dummy and linking atoms toward opposite directions on the *z*-axis (0.5 Å move at each step), simultaneously optimizing the geometry with applied constraints on the dummy and linking atoms. We repeat this step until there is a sudden bump in energy (set by the end-user), which is related to the sum of the dissociation energy of the backbone of a building block. We noticed that 50−150 kJ/mol energy change at a step occurs when the total energy of a molecule increases nearly 3−12% of the bond dissociation energy of all bonds in the backbone chain, which is sufficient for straightening a twisted molecule.

Finally, after being validated against the input SMILES string, the dummy atoms are removed, and the molecule is placed in a unit cell that is periodical along the axis connecting the dummy atoms (chosen to be the *z*-axis). Oligomers of length *n* are also generated by placing *n* molecules (with dummy atoms removed) along the *z*-axis, and the two end linking atoms are saturated with hydrogens. In contrast to `MoleculeBuilder`, repeating units in oligomer models generated using `ChainBuilder` are well organized in the *z*-direction. It is worth pointing out that, if there is a double bond between the linking and dummy atoms, oligomers are not generated, as hydrogens cannot be considered to satisfy the valency of linking atoms. In such a scenario, if there is a single bond in the polymer chain backbone, the user is encouraged to rewrite the SMILES so that a single bond connects the connecting and dummy atoms. In order for the oligomer/polymer chain to be isolated in further calculations using plane-wave codes like VASP,[36,37] the unit cell sizes along the periodic directions are selected so that the chain or oligomer is ≃12 Å apart from its periodic images. These oligomers/polymer chains are the output of PSP.

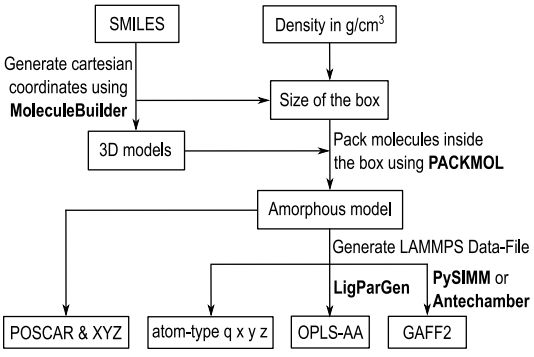
2.3. Small Molecule and Polymer Crystal Predictions. Crystal structures of small molecules or polymers may be predicted from an individual molecule or infinite chain using a simple but highly scalable procedure.[6] As described in Scheme 3, a copy of the oligomer/polymer chain, aligned along the *z*-direction, is placed far from the original chain, rotated by a given angle around itself, rotated by another angle around the original chain, translated along the *z*-direction by a given distance, and rigidly moved toward the original chain so that the distance between any two atoms is roughly but no less than 2 Å. This distance cutoff is large enough for not making unnecessary interchain bonds but small enough for further calculations to be effective. By specifying the number of samples for each rotating angle and translating distance, i.e., 3 degrees of freedom, a number of candidate crystal structures are obtained. In order to explore different crystal structures for small molecules, in addition to these movements, rotations of both chains around their *x*-, *y*-, and *z*-axes are considered, allowing for 8 degrees of freedom. These candidate structures are either provided as output or optionally optimized with UFF, and low-energy structures are chosen as the output of the package. As demonstrated in ref 6, by further optimizing these structures at the first-principles level of calculations, the ground-state structure of numerous linear polymers can be obtained.

2.4. Polymer Amorphous Structure Prediction. In order to provide a reasonable initial amorphous model for MD simulations, the workflow shown in Scheme 4 is implemented in the `AmorphousBuilder` module of PSP which requires SMILES strings and the density or size of a simulation box as inputs. At first, 3D models of oligomers are generated from SMILES using the `MoleculeBuilder`. Then, if the size of the simulation box is not provided, the volume of the box is estimated from the total molar mass of all molecules and the density (in units of g/cm^3) defined by the end-user. Based on the requirement of the shape of the box (i.e., cubic or rectangular), the lengths of the three axes are determined. In the next step, considering individual oligomers/molecules as rigid entities, they are translated and rotated randomly with a goal to minimize an objective function implemented in the PACKMOL package[11] that ensures all species are inside the simulation box with a user-defined intermolecular distance. The Cartesian coordinates of the system are extracted from the PACKMOL output, and an input file for LAMMPS is generated. If demanded by the user, an OPLS-AA parameter file is generated by producing parameters of individual molecules using LigParGen[14,38,39] and then appending them systematically for the entire model. It is worth noting that the LigParGen toolkit generates OPLS-AA parameters for neutral/charged organic molecules based on the CM1A[40] charge model. The generation of an OPLS-AA parameter file is limited to molecules with a maximum of 200 atoms, as LigParGen is incapable of handling larger compounds. Similar

Scheme 3. Schematic Workflow of the `CrystalBuilder` (CrB) Module



Scheme 4. Schematic Workflow of the `AmorphousBuilder` (AB) Module



https://doi.org/10.1021/acs.jctc.2c00022
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

to OPLS-AA, GAFF2 parameters for the system are also generated using PySIMM[29,30] or Antechamber.[41,42]

3. SOFTWARE DESCRIPTION

3.1. General Information. Polymer structure predictor PSP is a portable python toolkit for building atomic models of polymers from their SMILES strings. This open-source package is distributed under the MIT license and can be obtained from GitHub at https://github.com/Ramprasad-Group/PSP. To use this package, working installations of RDKit (v2020.03.3),[7] Open Babel (v3.1.1),[8] PACKMOL,[11] Numpy,[43] and Pandas[44] are needed. It is also necessary to have PySIMM,[29,30] LigParGen,[14,38,39] AmberTools21,[45] and LAMMPS[25] installed in order to use all of the functionalities of PSP. Unless otherwise specified, the default parameters recommended by these packages were used for molecular representation, energy calculation, geometry optimization, MD simulation, and some other purposes. All of the PSP modules, detailed documentation, and test examples can be accessed in *psp*, *documentation*, and *examples* directories, respectively. A setup script (*setup.py*) is provided for easy installation of the package, by simply running one command from a terminal using *python setup.py install*.

To make PSP more user-friendly, we created a Colab notebook that installs PSP and dependencies and provides several examples. By modifying input SMILES and parameters, users can build desired polymer models and download them for downstream *ab initio* or MD simulations. This notebook is distributed as a part of the PSP package.

Follow the steps below to install PSP and its dependencies on a local server:

(1) Install Anaconda/Miniconda and then install the following packages with conda
   - RDKit (v2020.03.3). For more details, see https://anaconda.org/rdkit/rdkit.
   - Open Babel (v3.1.1). For more details, see https://anaconda.org/conda-forge/openbabel.
   - AmberTools21. For more details, see https://anaconda.org/conda-forge/ambertools. Make sure to include the path for ANTECHAMBER executable as an environment variable "ANTECHAMBER_EXEC" in *.bashrc* file.

(2) Install PySIMM and LAMMPS and set PATHs as described at https://github.com/polysimtools/pysimm.

(3) Install PACKMOL and make sure to include the path for its executable as an environment variable "PACKMOL_EXEC" in *.bashrc* file.

(4) Install LigParGen dependencies: Obtain a copy of the BOSS[46,47] executable and set $BOSSdir variable in bash. For more information, see http://zarbi.chem.yale.edu/ligpargen and http://zarbi.chem.yale.edu/software.html. To make LigParGen compatible with PSP, we updated it to include the following features: (1) the ability to store output files in a user-defined directory and (2) compatibility with the recent versions of Open Babel (v3.1.1), NetworkX (v2.5), and pandas (v1.2.4). Note that NetworkX is not yet installed; this must be done prior to using PSP. The modified LigParGen code is redistributed as part of the PSP package.

3.2. Software Architecture. The main PSP library is composed of four classes, i.e., `MoleculeBuilder`, `ChainBuilder`, `CrystalBuilder`, and `AmorphousBuilder` (see Figure 1). As the names suggest, `MoleculeBuilder` and `ChainBuilder` take care of finite and infinite polymer chain construction, respectively. `CrystalBuilder` builds crystal models from oligomer/polymer chains, and `AmorphousBuilder` generates amorphous models from SMILES string and density or size of the simulation box. When PSP is installed, these modules can be imported and called in a simple manner. More details can be found in the next sections of this paper.

3.3. Software Functionalities. Polymer structure predictor PSP accepts SMILES as the input, returning oligomers, polymer chains, polymer crystals, and amorphous structures for further optimizations or MD simulations. The package allows us to set some parameters for keeping a balance between the computational cost and (possibly) the quality of polymer models. All output geometries are verified to ensure their correctness with respect to intramolecular and intermolecular connectivity. Output geometries for oligomers are provided in XYZ, PDB, and VASP POSCAR formats, while infinite polymer chains and crystals are exported in VASP POSCAR format. Amorphous models are exported as POSCAR and LAMMPS input formats. These outputs can then easily be converted to any other formats required by end users.

4. ILLUSTRATIVE EXAMPLES, VALIDATION, AND PERFORMANCE TEST

PSP provides preset parameters for novice users to quickly get polymer models by merely providing a polymer SMILES string. This section provides four tutorials for creating diverse polymer models, which are intended to give end-users systematic guidance for using PSP. Generated models were validated by comparing with experimentally known structures. The performance of this package was extensively tested using a Relion XE2112 server equipped with Intel Xeon SKL processors running the Ubuntu operating system.

4.1. Finite Polymer Chains. *Description.* Linear or loop chain models are required for physics-based simulations and the construction of amorphous structures. However, creating these 3D structures by hand is difficult and time-consuming. This tutorial will demonstrate how to effortlessly construct models for molecules and linear/loop oligomers using the `MoleculeBuilder` module.

Solution. Lets build models for 2-methoxypyridine-1,4-dimer (M1), acetonitrile oxide octamer (O1), and *t*-butyl(2-benzyloxycarbonyl)cyclopentyl-pentakis(2-carbamoylcyclopentyl)carbamate (O2). As depicted in Figure 2a, SMILES strings required for building these models are written in a CSV (comma-separated values) file.

(a)
```
ID, smiles, LeftCap, RightCap
M1, C13C=CC(C2C=CC1N=C2OC)N=C3OC,,
O1, C(C)(=NO[*])[*],,
O2, C1(C(CCC1)[*])NC(=O)[*], O(C(C)(C)C)[*], C(=O)(OCC1=CC=CC=C1)[*]
```

(b)
```
import pandas as pd
import psp.MoleculeBuilder as mb

input_df = pd.read_csv('molecule_input.csv')
mol = mb.Builder(input_df, ID_col='ID', SMILES_col='smiles',
      Length=[2, 5, 20, 50], Inter_Mol_Dis=6.0,
      NumConf=4, loop=False, OutDir='molecules_dir')
mol.Build()
```

Figure 2. Input files for `MoleculeBuilder`. (a) A CSV file (*molecule_input.csv*) defining SMILES strings of repeating, left-cap, and right-cap units. (b) Python script for building linear oligomers.

https://doi.org/10.1021/acs.jctc.2c00022
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

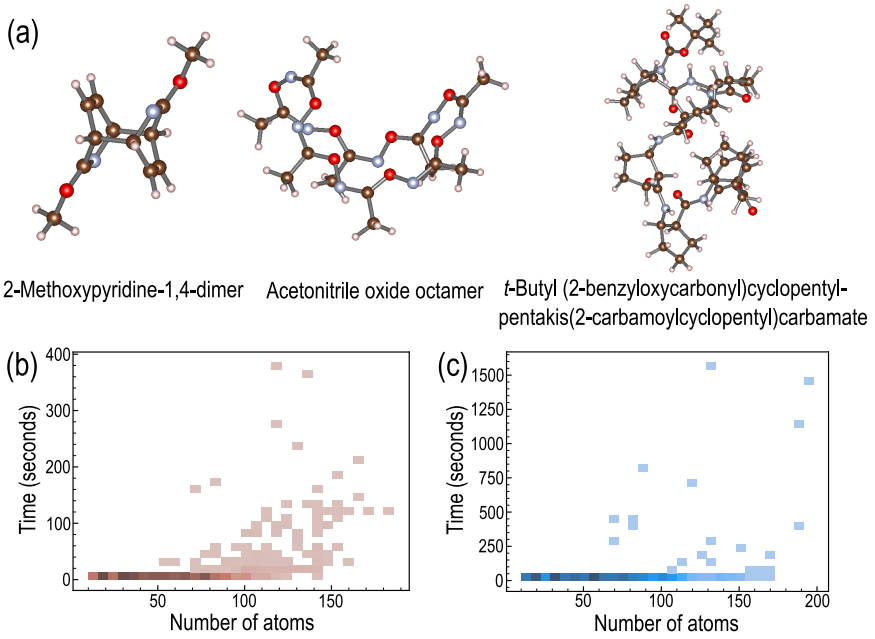


Figure 3. Molecular structures generated using `MoleculeBuilder` and performance test. (a) Geometry of three representative molecules, where carbon, hydrogen, nitrogen, and oxygen are shown in brown, light brown, cyan, and red colors, respectively. (b, c) Performance test for generating dimer models from 1000 SMILES strings using single-core and multicore processors for individual molecules.

It is supplied as a pandas Dataframe to `MoleculeBuilder`. Figure 2b shows an example of a python code snippet required for this module. Here, *df_smiles* is a pandas Dataframe whose two columns *ID_col* and *SMILES_col* refer to identity name and SMILES of the repeating unit of a polymer, respectively. *Length* is a python list of integers determining the length of an oligomer. *Inter_Mol_Dis* stands for the intermolecular distance between two molecules in adjacent simulation boxes in *x*-, *y*-, and *z*-directions. *NumConf* is an integer defining the number of output conformers for a molecule. *Loop* is a python boolean, where *True* stands for circular models, and otherwise, a linear chain model will be generated. *OutDir* is the PATH of a directory to store output files.

Models for three systems (M1, O1, and O2) were generated by defining the length of oligomers and their forms (linear or loop) in the code snippet shown in Figure 2b. To validate models, they were compared with experiments, which were found to be sufficiently accurate for use as the starting geometry for high-level computations. For example, the geometries of M1,[48] O1 (curved structure),[49] and O2 (helical geometry)[50] are close to experimental data, as illustrated in Figure 3a. In fact, the ability of RDKit to generate experimentally determined structures is extensively documented in ref 51. Users interested in getting the most stable conformer or a pool of conformers for an individual molecule/oligomer are encouraged to generate multiple conformers by specifying *NumConf* as mentioned before, followed by geometry optimization using high-level *ab initio* methods.

Additionally, ring-opening polymerization enthalpy (ΔH) was computed for hundreds of systems starting from monomer and loop oligomer models. Ring-opening polymerization is a process during which a cationic or anionic terminal group of a polymer chain opens a cyclic monomer and incorporates it into the chain. ΔH is a thermodynamic parameter indicating how easily the rings can be opened to initiate the polymerization. Results from this newly developed scheme, which significantly surpasses the current approach in correctly computing the ΔH (against experimental data), will soon be published separately.[52]

Performance Test. The performance of `MoleculeBuilder` was verified with 1000 distinct polymer SMILES on one-core and multicore CPUs. It should be noted that, for multicore processors, the system's maximum supported cores were utilized with a total of 1,000,000 attempts for the conformer search, but for one core processor, just 10,000 attempts were considered to minimize the computational cost. Figure 3b,c depicts the number of dimer models successfully generated while preserving atomic connectivity information encoded in SMILES strings. It shows that the success rates for single-core and multicore processors are 97.2 and 98.5%, respectively. According to Figure 3c, 93.5% of models generated using multicore CPUs require less than 15 s to execute, which includes oligomers >150 atoms. Notably, conformer search is a tough and delicate task, especially for large and branched oligomers, as it is required to examine a wide search space due to the large number of rotatable bonds and intramolecular conflicts caused by branches, resulting in a high computing cost or even failure.

4.2. Infinite Polymer Chains. *Description.* The periodic structure of a polymer represents an infinite chain that is used as an input for *ab initio* simulation and for creating polymer crystal structures. Building such a model is particularly challenging, as no packages known to date can build them. Here, we will provide several examples to build infinite chain models.

Solution. Let us build infinite chain models of three well-known polymers: polyethylene, polyvinyl chloride, and polyvinylidene fluoride (PVDF). The input SMILES strings required for building them are provided in a CSV file, as shown in Figure 4a.

`ChainBuilder` is called to predict polymer chain structures (Figure 4b). Although most of the parameters are predefined, users can tune them as necessary. Here, *df_smiles* is

https://doi.org/10.1021/acs.jctc.2c00022
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

(a)
```
ID, smiles
PE, [*]CC[*]
Isotactic_PVC, C(C([*])Cl)[*]
Syndiotactic_PVC, C(C(CC([*])Cl)Cl)[*]
PVDF, C(C(F)(F)[*])[*]
```

(b)
```
import pandas as pd
import psp.ChainBuilder as ChB

df_smiles = pd.read_csv("chain.csv")
chain_builder = ChB.Builder(df_smiles, ID_col='ID',
                SMILE_col='smiles', Length=[1,2,5,'n'],
                Method='SA', Steps=25, Substeps=20,
                IntraChainCorr=1, Tol_ChainCorr=50,
                OutDir='chain-models', NCores=4)
results = chain_builder.BuildChain()
```

Figure 4. Input files for `ChainBuilder`. (a) A CSV file (*chain.csv*) defining SMILES strings of repeating units. (b) Python script for building oligomers and infinite polymer chain structures.

a pandas Dataframe of input polymer SMILES strings, whose two columns are specified by *ID_col* and *SMILES_col*, respectively. The length of the oligomers and polymer chains is given by a finite integer or *n* in *Length*, a python list. The method defines the algorithm to be used; the simulated annealing approach is preset and highly recommended. Steps and substeps are, respectively, the number of outer and inner loops of the simulated annealing algorithm; the former governs the "cooling down" process of the system, while the latter manages the trial moves away from the current position. With an increase in the number of steps and substeps, the success rate of getting polymer chains, as well as the quality of chains, will improve but with a higher computational cost. A twisted polymer backbone can be straightened by setting *IntraChainCorr* = *True* with a cutoff for a sudden energy bump defined by *Tol_ChainCorr*. `ChainBuilder` supports parallel computation, and the number of cores can be defined by *NCores*.

Generated chain models were validated by comparing them with experimentally known structures. The polymer chain models predicted of PE, PVC, and PVDF, shown in Figure 5, are acceptable. In particular, the predicted chain structure of PE is identical with the experimental data.[53] Moreover, both the isotactic and syndiotactic conformations of PVC[54] were obtained using PSP. In the case of PVDF,[55−57] we always get a β-phase chain out of five phases known (α, β, γ, δ, η), although the α phase is readily accessible in experiment. The reason is that, at the UFF level of energy computations, the chain conformation of the β phase (E_{UFF} = 48.5 kJ/mol) is significantly more stable than that of the α phase (E_{UFF} = 69.5 kJ/mol). Clearly, the capability of PSP is limited by UFF, the classical potential used for the atomic structure optimizations. We also note that the "correct" conformation of a polymer may be a bulk property; i.e., it is determined not only by the chain energy but also by the surrounding environment in bulk polymers.[6] Thus, by predicting a "stand-alone" polymer chain structure, we have adopted an approximation that may limit the accuracy of PSP.

Apart from comparing geometries, we used infinite chain models to calculate the bandgap and charge injection barriers for over 4000 polymers.[58−61] As there is close agreement between computed and experimental properties, infinite chain models can be considered suitable for downstream *ab initio* computations. All of these data are incorporated into the Polymer Genome and Khazana, which are freely accessible to the research community.

Performance Test. The performance of infinite polymer chain generation was evaluated using a 1000 distinct polymer SMILES data set and various methods implemented in `ChainBuilder` using a single core per model. As seen in Figure 5b, simple dimerization of two monomers results in a success rate of 71.8%, which is significantly improved to 94.6% for SA with stretching of the repeating unit (threshold = 150

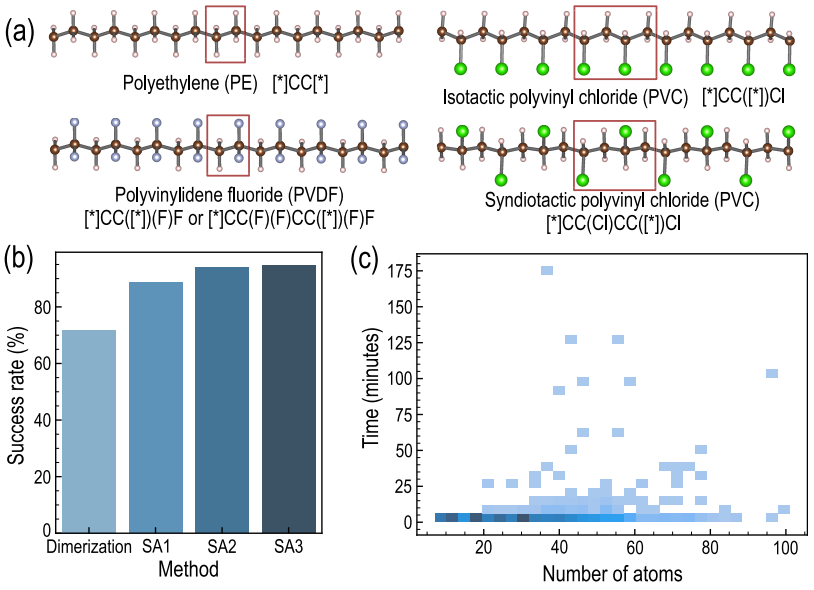


Figure 5. Polymer chain models generated using `ChainBuilder` and performance test. (a) Chain structure of PE, PVC, and PVDF, where the red box indicates the repeating unit. Polymer SMILES used as input for `ChainBuilder` are also provided. Carbon, hydrogen, nitrogen, and chlorine are shown in brown, light brown, cyan, and green colors, respectively. (b) Success rates for generating polymer chain models using different methods implemented in `ChainBuilder`, including dimerization and simulated annealing without stretching (SA1), stretching with a threshold of 50 kJ/mol (SA2), and stretching with a threshold of 150 kJ/mol (SA3). (c) Time required to generate each model from SMILES using a single core.

https://doi.org/10.1021/acs.jctc.2c00022
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

kJ/mol). Note that, in the dimerization process, we keep the first monomer aligned along the *z*-axis and rotate the second monomer around its *z*-axis in order to find a suitable conformer for a dimer to which boundary conditions can be applied. It is worth mentioning that models created with the SA method have higher energetic stability and demand less computing time for high-level calculations due to the reduced number of atoms in a unit cell. The computational cost for building an infinite chain model depends on many factors such as (1) the number of atoms in a unit cell, (2) the number of rotatable bonds, and (3) the number of steps in the inner and outer loops for the SA method. The computational cost required for each model for SA2 (steps = 100 and substeps = 20) is depicted in Figure 7c. As can be seen, 81.4% of samples require <5 min, in which 57.9% require <1 min.

4.3. Crystal Structures. *Description.* The generation of possible crystal structures is one of the most difficult challenges in predicting crystal structures and computing their properties. This tutorial shows how `CrystalBuilder` builds a pool of candidate crystal models from individual polymer chains or small molecules. Individual models can be generated using `MoleculeBuilder/ChainBuilder` or provided separately by the user. It is worth noting that input chain models are expected to be aligned along their *z*-axis.

Solution. `CrystalBuilder` requires only an input of the desired polymer chain in the POSCAR file format. In the following example, we will build crystal models using the PVC chain structure generated by `ChainBuilder` in the previous tutorial and the code snippet depicted in Figure 6.

```
import psp.CrystalBuilder as CrB

chain_list = ["chains/Syndiotactic_PVC/Syndiotactic_PVC.vasp"]
crystal_builder = CrB.Builder(VASPInp_list = chain_list,
                  NSamples=5, InputRadius='auto', MinAtomicDis=2.0,
                  Polymer=True, OutDir='crystals-models')
results = crystal_builder.BuildCrystal()
```

Figure 6. Python script for building crystal models from a polymer chain.

In the python script, *chain_list* refers to a list of input file names, i.e., the name of the PVC chain model present in the chains directory. Moreover, *NSamples* is, as mentioned in section 2.3, the number of samples for each degree of freedom, thus determining the maximum number of possible crystal candidates, which is n^3 for infinite polymer chains (*Polymer* = *True*) and n^8 for small molecules. *InputRadius* is the distance between the *z*-axes of two chains, and it can be set to either *Auto* or any desired value. If *Auto* is placed, `CrystalBuilder` approximately calculates the smallest distance suitable for all possible crystal models, considering *x* and *y* coordinates of the input geometry and the minimum atomic distance defined by *MinAtomicDis*. To avoid undesired interaction between polymer chains, the interatomic distance between any two atoms of two chains is about 2 Å and above. A few snapshots of crystal models for syndiotactic PVC are shown in Figure 7. These crystal models can be optimized using UFF and downselected low-energy structures by specifying *Optimize* = *True* and *NumCandidate* = *N*, where *N* is an integer. *NumCandidate* indicates the number of candidates to choose and is returned by PSP as an output. Crystal candidates can be further studied using *ab initio* or MD simulations. Using this approach, known crystal structures of PE, PVC, PVDF, and three other linear polymers were recovered,[6] along with some other energetically competing phases. In addition, bandgaps for 300 crystal structures were computed and incorporated into Polymer Genome.

Performance Test. The creation of crystal models from polymer chains is considerably fast. For example, using a single-core CPU, it takes 0.91, 1.88, and 10.45 s to build 27, 125, and 1000 crystal models for PVC, respectively. Optimizations of models are optional and incur a computational cost proportional to the number of atoms in a unit cell.

4.4. Amorphous Structures. *Description.* Constructing an amorphous model for polymers/small molecules is not a trivial task. It requires packing all of the molecules in a simulation box while keeping enough intermolecular distance to avoid unwanted interactions. This tutorial demonstrates how `AmorphousBuilder` builds an amorphous model for a polymer using the repeated unit SMILES.

Solution. To generate an amorphous structure, `AmorphousBuilder` requires information on molecules/oligomers as well as the simulation box. All information regarding molecules/oligomers to be included in an amorphous model can be provided in a CSV file. In the following example, we will build an amorphous model for the PE system. In the simulation box, we included 100 PEs with a chain length of 25 monomers for each, that can be defined in a CSV file as depicted in Figure 8a.

The code snippet shown in Figure 8b is an example for building an amorphous model. Here, *input_df* is a Pandas

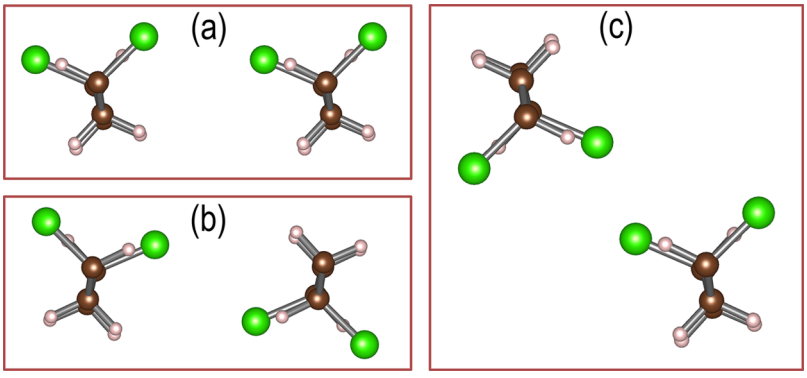


Figure 7. Examples of crystal models (top views) for syndiotactic PVC, in which one chain is rotated around its axis as well as with respect to the other chain, keeping the latter one fixed. The periodic boundary conditions specified along the *x*- and *y*-directions are represented as a box. Carbon, hydrogen, and chlorine are shown in brown, light brown, and green colors, respectively.

https://doi.org/10.1021/acs.jctc.2c00022
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

(a)
```
ID,smiles,Len,Num,NumConf,Loop
CC25,[*]CC[*],25,100,1,False
```

(b)
```
import pandas as pd
import psp.AmorphousBuilder as ab

input_df = pd.read_csv('input_amor.csv')
amor = ab.Builder(input_df, ID_col='ID', SMILES_col='smiles',
       Length='Len', NumMole='Num', NumConf='NumConf',
       LeftCap='LeftCap', RightCap='RightCap', Loop='Loop',
       density=0.65, tol_dis=2.0, box_type='c',
      OutFile='amor_model', OutDir='amorphous_models')
amor.Build()
```

Figure 8. Input files for `AmorphousBuilder`. (a) A CSV file (*input_amor.csv*) defining SMILES strings of repeating units, length of oligomers, number of oligomers to be included in the amorphous model, etc. Users can add more molecules/oligomers to the amorphous model by appending new raws to the list. (b) Python script for building an amorphous model.

Dataframe for defining molecules/oligomers in an amorphous model, in which column names for identity name, SMILES, length of an oligomer, the number of molecules, the number of conformers, left-cap SMILES, right-cap SMILES, and shape of oligomer chain (i.e., linear or loop) are specified by *ID_col*, *SMILES_col*, *Length*, *NumMole*, *NumConf*, *LeftCap*, *RightCap*, and *Loop*, respectively. The density of the simulation box is defined by *density* (g/cm^3). *tol_dis* stands for the minimum intermolecular distance in Å. *box_type* determines the shape of the simulation box, where *c* and *r* denote cubic and rectangular shape, respectively. *OutDir* indicates the PATH of the output directory, and *OutFile* is the name of the output file generated by `AmorphousBuilder`.

The amorphous model generated for the PE system is shown in Figure 9a. To ensure PSP-generated amorphous models are good enough for MD simulation, we computed the glass transition temperature (T_g) for the PE system. First, the OPLS force field parameters were obtained for individual molecules using LigParGen[14,38,39] and combined systematically using an in-house script. Second, the system was equilibrated following the 21-step equilibration procedure suggested by Abbott et al.,[62] allowing for efficient relaxation and compression of the structure. Then, for each 20 K, the system was equilibrated for 1 ns using LAMMPS,[25] and the variation of density with respect to the temperature is depicted in Figure 9b. As can be seen, the estimated T_g (241 K) is within the experimental T_g range and is close to those computed using MD simulations in recent studies.[63,64]

Performance Test. The overall time required to construct an amorphous model depends on the computational costs associated with building oligomers using `Molecule-Builder` (section 4.1) and packing molecules by the PACKMOL package.[11] The PE model seen in Figure 9a, containing 15,200 atoms, required 11.62 min. It is worth noting that the computational costs for packing molecules increases with an increase in the number of atoms and density of the simulation box, as it is challenging to minimize the objective function implemented in the PACKMOL package.

Each PSP builder supports a number of additional parameters or keywords, which are detailed in the user manual. The provided notebook contains a plethora of additional examples for creating polymer models. Users are encouraged to go through the documents for more detailed information.

5. LIMITATIONS AND FUTURE DIRECTIONS

Although the current version of PSP addresses several issues and is capable of building a hierarchy of polymer models using SMILES strings, there are certain limitations. For example, building models for highly branched and large oligomers using the `MoleculeBuilder` module is still challenging due to the large number of rotatable bonds and intramolecular conflicts caused by branches. As a result, they require additional computational time and occasionally fail to generate an acceptable initial model. Additionally, `Molecule-Builder` does not ensure that the conformational ensemble contains all possible important conformers of a molecule or an oligomer.

PSP currently supports homopolymers but not copolymers[65−69] such as (a) alternating copolymers, (b) block copolymers, (c) random copolymers, and (d) gradient copolymers. Polymers with a ladder-like structure[70,71] and polymer networks[72,73] are also not included. These polymer classes are in high demand and are used in a wide range of applications. Furthermore, it would be beneficial to generate input files that include parameters particular to various computational packages, allowing direct execution of physics-based simulations. Thus, we aim to keep developing the PSP package in order to include all of these aspects in the future edition.

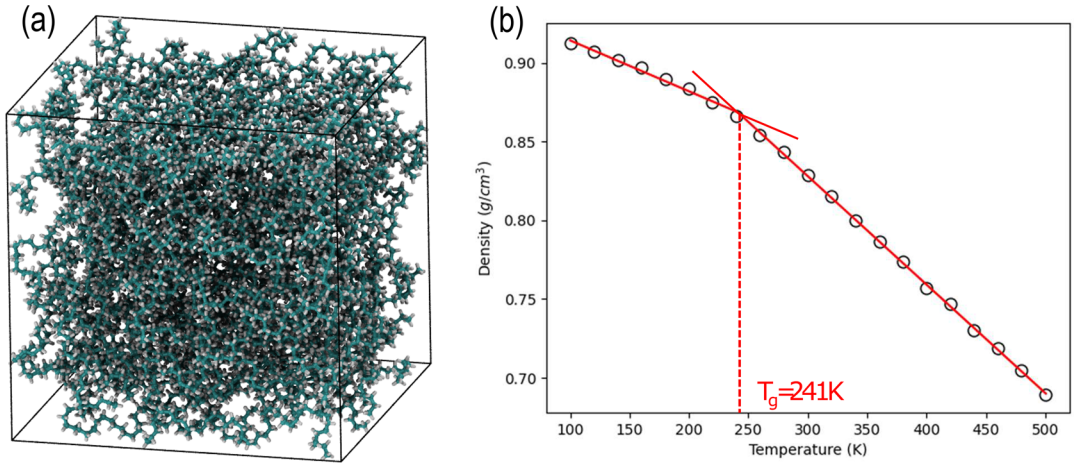


Figure 9. Glass transition temperature of the polyethylene (PE) system. (a) Amorphous model for 100 PEs with a chain length of 25 monomers and (b) variations of density with respect to the temperature. For each 20 K, the system was equilibrated for 1 ns.

https://doi.org/10.1021/acs.jctc.2c00022
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

6. SUMMARY

We have developed and released a python toolkit named polymer structure predictor (PSP) for predicting a hierarchy of atomic polymer models, i.e., oligomers, polymer chains, crystal structures, and amorphous models, starting from a SMILES string of the polymer repeating unit. GAFF2 and OPLS-AA parameter files for downstream LAMMPS MD simulations and structure files for downstream *ab initio* calculations are also provided. The flexibility to define parameters allows for the construction of desired models at a reasonable computational cost. The toolkit may easily be integrated in automated software pipelines. The performance of this package is tested by comparing models with the known experimental data for several polymers. With the help of PSP, databases of bandgaps and charge injection barriers for several thousands of polymers were created, which are provided in Khazana (https://khazana.gatech.edu) and contributed to the freely accessible Polymer Genome platform (www.polymergenome.org) designed for the near-instantaneous prediction of a variety of polymer properties. The PSP package includes a Colab notebook, providing a set of self-contained instructions for installation and examples for building polymer models, which can be visualized and downloaded for later use.

The PSP package provides an automated platform to generate 3D models of polymers, a starting point for *in silico* simulations. This functionality is not only essential for establishing an autonomous polymer discovery platform like CAMD but also essential for the majority of atomic-level polymer simulations. For experimental polymer scientists, PSP could be useful for making simple visual models of polymers without requiring specialized computational infrastructures. In the context of the emerging polymer informatics ecosystem, this toolkit will substantially reduce efforts to develop extensive databases of computed polymer properties. PSP is expected to be beneficial for a wide number of academic and industrial research activities, enabling automation in polymer simulation and computational data generation.

AUTHOR INFORMATION

Corresponding Authors

Huan Tran − *School of Materials Science and Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332, United States;* orcid.org/0000-0002-8093-9426; Email: huan.tran@mse.gatech.edu

Rampi Ramprasad − *School of Materials Science and Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332, United States;* orcid.org/0000-0003-4630-1565; Email: rampi.ramprasad@mse.gatech.edu

Authors

Harikrishna Sahu − *School of Materials Science and Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332, United States;* orcid.org/0000-0001-5458-9488

Kuan-Hsuan Shen − *School of Materials Science and Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332, United States*

Joseph H. Montoya − *Accelerated Materials Design and Discovery, Toyota Research Institute, Los Altos, California 94022, United States;* orcid.org/0000-0001-5760-2860

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jctc.2c00022

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

This work is financially supported by Toyota Research Institute through the Accelerated Materials Design and Discovery program.

REFERENCES

(1) Jayaraman, A. 100th Anniversary of Macromolecular Science Viewpoint: Modeling and Simulation of Macromolecules with Hydrogen Bonds: Challenges, Successes, and Opportunities. *ACS Macro Lett.* **2020**, *9*, 656−665.

(2) Zhai, C.; Li, T.; Shi, H.; Yeo, J. Discovery and Design of Soft Polymeric Bio-Inspired Materials with Multiscale Simulations and Artificial Intelligence. *J. Mater. Chem. B* **2020**, *8*, 6562−6587.

(3) Gartner, T. E.; Jayaraman, A. Modeling and Simulations of Polymers: A Roadmap. *Macromolecules* **2019**, *52*, 755−786.

(4) Huan, T. D.; Boggs, S.; Teyssedre, G.; Laurent, C.; Cakmak, M.; Kumar, S.; Ramprasad, R. Advanced Polymeric Dielectrics for High Energy Density Applications. *Prog. Mater. Sci.* **2016**, *83*, 236.

(5) Mannodi-Kanakkithodi, A.; Treich, G.; Huan, T. D.; Ma, R.; Tefferi, M.; Cao, Y.; Sotzing, G.; Ramprasad, R. Rational Co-Design of Polymer Dielectrics for Energy Storage. *Adv. Mater.* **2016**, *28*, 6277−6291.

(6) Huan, T. D.; Ramprasad, R. Polymer Structure Prediction from First Principles. *J. Phys. Chem. Lett.* **2020**, *11*, 5823−5829.

(7) *RDKit: Cheminformatics and Machine Learning Software*; Open-source, 2018; http://www.rdkit.org.

(8) O'Boyle, N. M.; Banck, M.; James, C. A.; Morley, C.; Vandermeersch, T.; Hutchison, G. R. Open Babel: An Open Chemical Toolbox. *J. Cheminformatics* **2011**, *3*, 33.

(9) Turcani, L.; Berardo, E.; Jelfs, K. E. stk: A Python Toolkit for Supramolecular Assembly. *J. Comput. Chem.* **2018**, *39*, 1931−1942.

(10) Mishra, A.; Chen, L.; Li, Z.; Nomura, K.; Krishnamoorthy, A.; Fukushima, S.; Tiwari, S. C.; Kalia, R. K.; Nakano, A.; Ramprasad, R.; Sotzing, G.; Cao, Y.; Shimojo, F.; Vashishta, P. Computational Framework for Polymer Synthesis to Study Dielectric Properties using Polarizable Reactive Molecular Dynamics. 2020, arXiv:2011.09571v1 [physics.chem-ph]. arXiv.org e-Print archive. https://arxiv.org/abs/2011.09571.

(11) Martínez, L.; Andrade, R.; Birgin, E. G.; Martínez, J. M. PACKMOL: A Package for Building Initial Configurations for Molecular Dynamics Simulations. *J. Comput. Chem.* **2009**, *30*, 2157−2164.

(12) Martínez, J. M.; Martínez, L. Packing Optimization for Automated Generation of Complex System's Initial Configurations for Molecular Dynamics and Docking. *J. Comput. Chem.* **2003**, *24*, 819−825.

(13) Ramos, M. C.; Quoika, P. K.; Horta, V. A. C.; Dias, D. M.; Costa, E. G.; do Amaral, J. L. M.; Ribeiro, L. M.; Liedl, K. R.; Horta, B. A. C. pyPolyBuilder: Automated Preparation of Molecular Topologies and Initial Configurations for Molecular Dynamics Simulations of Arbitrary Supramolecules. *J. Chem. Inf. Model.* **2021**, *61*, 1539−1544.

(14) Dodda, L. S.; Vilseck, J. Z.; Tirado-Rives, J.; Jorgensen, W. L. 1.14*CM1A-LBCC: Localized Bond-Charge Corrected CM1A Charges for Condensed-Phase Simulations. *J. Phys. Chem. B* **2017**, *121*, 3864−3870.

(15) Choi, Y. K.; Park, S.-J.; Park, S.; Kim, S.; Kern, N. R.; Lee, J.; Im, W. CHARMM-GUI Polymer Builder for Modeling and Simulation of Synthetic Polymers. *J. Chem. Theory Comput.* **2021**, *17*, 2431−2443.

(16) Jo, S.; Kim, T.; Iyer, V. G.; Im, W. CHARMM-GUI: A web-based graphical user interface for CHARMM. *J. Comput. Chem.* **2008**, *29*, 1859−1865.

(17) Weininger, D. SMILES, a Chemical Language and Information System. 1. Introduction to Methodology and Encoding Rules. *J. Chem. Inf. Comput. Sci.* **1988**, *28*, 31−36.
(18) Momma, K.; Izumi, F. VESTA: A Three-Dimensional Visualization System for Electronic and Structural Analysis. *J. Appl. Crystallogr.* **2008**, *41*, 653−658.
(19) Huan, T. D.; Sahu, H.; Gonzalez Del Rio, B.; Aykol, M.; Montoya, J. H.; Kwon, H.-K.; Kamal, D.; Storeya, B.; Ramprasad, R. Autonomous Artificial-Intelligent Based Agents for Computational Polymer Discovery. Manuscript in preparation, 2022.
(20) Montoya, J. H.; Winther, K. T.; Flores, R. A.; Bligaard, T.; Hummelshøj, J. S.; Aykol, M. Autonomous Intelligent Agents for Accelerated Materials Discovery. *Chem. Sci.* **2020**, *11*, 8517−8532.
(21) Rappe, A. K.; Casewit, C. J.; Colwell, K. S.; Goddard, W. A.; Skiff, W. M. UFF, a Full Periodic Table Force Field for Molecular Mechanics and Molecular Dynamics Simulations. *J. Am. Chem. Soc.* **1992**, *114*, 10024−10035.
(22) Kresse, G.; Furthmüller, J. Efficiency of *ab-initio* Total Energy Calculations for Metals and Semiconductors Using a Plane-Wave Basis Set. *Comput. Mater. Sci.* **1996**, *6*, 15−50.
(23) Neese, F. Software Update: the ORCA Program System, Version 4.0. *WIREs Comput. Mol. Sci.* **2018**, *8*, No. e1327.
(24) Barca, G. M. J.; Bertoni, C.; Carrington, L.; Datta, D.; De Silva, N.; Deustua, J. E.; Fedorov, D. G.; Gour, J. R.; Gunina, A. O.; Guidez, E.; Harville, T.; Irle, S.; Ivanic, J.; Kowalski, K.; Leang, S. S.; Li, H.; Li, W.; Lutz, J. J.; Magoulas, I.; Mato, J.; Mironov, V.; Nakata, H.; Pham, B. Q.; Piecuch, P.; Poole, D.; Pruitt, S. R.; Rendell, A. P.; Roskop, L. B.; Ruedenberg, K.; Sattasathuchana, T.; Schmidt, M. W.; Shen, J.; Slipchenko, L.; Sosonkina, M.; Sundriyal, V.; Tiwari, A.; Galvez Vallejo, J. L.; Westheimer, B.; Wloch, M.; Xu, P.; Zahariev, F.; Gordon, M. S. Recent Developments in the General Atomic and Molecular Electronic Structure System. *J. Chem. Phys.* **2020**, *152*, 154102.
(25) Plimpton, S. J. Fast Parallel Algorithms for Short-Range Molecular Dynamics. *J. Comput. Phys.* **1995**, *117*, 1−19.
(26) Robertson, M. J.; Tirado-Rives, J.; Jorgensen, W. L. Improved Peptide and Protein Torsional Energetics with the OPLS-AA Force Field. *J. Chem. Theory Comput.* **2015**, *11*, 3499−3509.
(27) GAFF and GAFF2 are public domain force fields and are part of the AmberTools16 distribution, available for download at https://ambermd.org. According to the AMBER development team, the improved version of GAFF, GAFF2, is an ongoing project aimed at "reproducing both the high quality interaction energies and key liquid properties such as density, heat of vaporization and hydration free energy". GAFF2 is expected "to be an even more successful general purpose force field" and that "GAFF2-based scoring functions will significantly improve the successful rate of virtual screenings".
(28) Doan Tran, H.; Kim, C.; Chen, L.; Chandrasekaran, A.; Batra, R.; Venkatram, S.; Kamal, D.; Lightstone, J. P.; Gurnani, R.; Shetty, P.; Ramprasad, M.; Laws, J.; Shelton, M.; Ramprasad, R. Machine-Learning Predictions of Polymer Properties with Polymer Genome. *J. Appl. Phys.* **2020**, *128*, 171104.
(29) Fortunato, M. E.; Colina, C. M. *PySIMM*; https://github.com/polysimtools/pysimm.
(30) Fortunato, M. E.; Colina, C. M. pysimm: A Python Package for Simulation of Molecular Systems. *SoftwareX* **2017**, *6*, 7−12.
(31) Vollhardt, K. P. C.; Schore, N. E. *Organic Chemistry; Palgrave Version: Structure and Function*; MacMillan International Higher Education: 2014.
(32) McMurry, J. *Organic Chemistry*, 8th ed.; Brooks/Cole Cengage Learning: 2012.
(33) Murcko, M. A.; Castejon, H.; Wiberg, K. B. Carbon-Carbon Rotational Barriers in Butane, 1-Butene, and 1, 3-Butadiene. *J. Phys. Chem.* **1996**, *100*, 16162−16168.
(34) Dragojlovic, V. Conformational Analysis of Cycloalkanes. *ChemTexts* **2015**, *1*, 14.
(35) Kirkpatrick, S.; Gelatt, C. D.; Vecchi, M. P. Optimization by Simulated Annealing. *Science* **1983**, *220*, 671−680.
(36) Kresse, G.; Furthmüller, J. Efficiency of *ab-initio* Total Energy Calculations for Metals and Semiconductors Using a Plane-Wave Basis Set. *Comput. Mater. Sci.* **1996**, *6*, 15−50.
(37) Kresse, G.; Furthmüller, J. Efficient Iterative Schemes for *ab-initio* Total-Energy Calculations Using a Plane-Wave Basis Set. *Phys. Rev. B* **1996**, *54*, 11169−11186.
(38) Dodda, L. S.; Cabeza de Vaca, I.; Tirado-Rives, J.; Jorgensen, W. L. LigParGen Web Server: An Automatic OPLS-AA Parameter Generator for Organic Ligands. *Nucleic Acids Res.* **2017**, *45*, W331−W336.
(39) Jorgensen, W. L.; Tirado-Rives, J. Potential Energy Functions for Atomic-Level Simulations of Water and Organic and Biomolecular Systems. *Proc. Natl. Acad. Sci. U. S. A.* **2005**, *102*, 6665−6670.
(40) Udier-Blagović, M.; Morales De Tirado, P.; Pearlman, S. A.; Jorgensen, W. L. Accuracy of Free Energies of Hydration Using CM1 and CM3 Atomic Charges. *J. Comput. Chem.* **2004**, *25*, 1322−1332.
(41) Wang, J.; Wang, W.; Kollman, P. A.; Case, D. A. Automatic Atom Type and Bond Type Perception in Molecular Mechanical Calculations. *J. Mol. Graph. Model.* **2006**, *25*, 247−260.
(42) Wang, J.; Wolf, R. M.; Caldwell, J. W.; Kollman, P. A.; Case, D. A. Development and Testing of a General Amber Force Field. *J. Comput. Chem.* **2004**, *25*, 1157−1174.
(43) Harris, C. R.; Millman, K. J.; van der Walt, S. J.; Gommers, R.; Virtanen, P.; Cournapeau, D.; Wieser, E.; Taylor, J.; Berg, S.; Smith, N. J.; Kern, R.; Picus, M.; Hoyer, S.; van Kerkwijk, M. H.; Brett, M.; Haldane, A.; del Río, J. F.; Wiebe, M.; Peterson, P.; Gérard-Marchant, P.; Sheppard, K.; Reddy, T.; Weckesser, W.; Abbasi, H.; Gohlke, C.; Oliphant, T. E. Array Programming with NumPy. *Nature* **2020**, *585*, 357−362.
(44) The pandas development team, *pandas-dev/pandas: Pandas*; Zenodo, 2020; DOI: 10.5281/zenodo.3509134.
(45) Case, D. A.; Aktulga, H. M.; Belfon, K.; Ben-Shalom, I. Y.; Brozell, S. R.; Cerutti, D. S.; Cheatham, T. E., III; Cruzeiro, V. W. D.; Darden, T. A.; Duke, R. E.; Giambasu, G.; Gilson, M. K.; Gohlke, H.; Goetz, A. W.; Harris, R.; Izadi, S.; Izmailov, S. A.; Jin, C.; Kasavajhala, K.; Kaymak, M. C.; King, E.; Kovalenko, A.; Kurtzman, T.; Lee, T. S.; LeGrand, S.; Li, P.; Lin, C.; Liu, J.; Luchko, T.; Luo, R.; Machado, M.; Man, V.; Manathunga, M.; Merz, K. M.; Miao, Y.; Mikhailovskii, O.; Monard, G.; Nguyen, H.; O'Hearn, K. A.; Onufriev, A.; Pan, F.; Pantano, S.; Qi, R.; Rahnamoun, A.; Roe, D. R.; Roitberg, A.; Sagui, C.; Schott-Verdugo, S.; Shen, J.; Simmerling, C. L.; Skrynnikov, N. R.; Smith, J.; Swails, J.; Walker, R. C.; Wang, J.; Wei, H.; Wolf, R. M.; Wu, X.; Xue, Y.; York, D. M.; Zhao, S.; Kollman, P. A. *Amber 2021*; University of California: San Francisco, CA, 2021.
(46) Jorgensen, W. L.; Tirado-Rives, J. Molecular Modeling of Organic and Biomolecular Systems Using BOSS and MCPRO. *J. Comput. Chem.* **2005**, *26*, 1689−1700.
(47) Jorgensen, W. L. In *The Encyclopedia of Computational Chemistry*; Schleyer, P. v. R., Ed.; John Wiley & Sons Ltd: Athens, NY, 1998; Vol. *5*, pp 3281−3285.
(48) Pridgen, L. N.; Chodosh, D. F.; Poziomek, E. J. The 1,4 Dimer of 2-Methoxypyridine $C_{12}H_{14}N_2O_2$. *Cryst. Struct. Commun.* **1981**, *10*, 1479.
(49) Menchetti, S.; Sabelli, C. Crystal and Molecular Structure of an Octamer of Acetonitrile Oxide. *J. Chem. Soc., Perkin Trans.* **1977**, *2*, 334−336.
(50) Appella, D. H.; Christianson, L. A.; Klein, D. A.; Richards, M. R.; Powell, D. R.; Gellman, S. H. Synthesis and Structural Characterization of Helix-Forming β-Peptides: trans-2-Aminocyclopentanecarboxylic Acid Oligomers. *J. Am. Chem. Soc.* **1999**, *121*, 7574−7581.
(51) Ebejer, J.-P.; Morris, G. M.; Deane, C. M. Freely Available Conformer Generation Methods: How Good Are They? *J. Chem. Inf. Model.* **2012**, *52*, 1146−1158.
(52) Tran, H.; Toland, A.; Stellmach, K.; Gutekunst, W.; Ramprasad, R. A Powerful and Versatile Scheme for Computing Ring-Opening Polymerization Enthalpy from First Principles. Manuscript in preparation, 2022.

https://doi.org/10.1021/acs.jctc.2c00022
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

(53) Peacock, A. *Handbook of Polyethylene: Structures: Properties, and Applications*, 1st ed.; CRC Press: New York, 2000.
(54) Wilkes, C. E.; Folt, V. L.; Krimm, S. Crystal Structure of Poly(vinyl chloride) Single Crystals. *Macromolecules* **1973**, *6*, 235−237.
(55) Ruan, L.; Yao, X.; Chang, Y.; Zhou, L.; Qin, G.; Zhang, X. Properties and Applications of the β Phase Poly(vinylidene fluoride). *Polymers* **2018**, *10*, 228.
(56) Hasegawa, R.; Kobayashi, M.; Tadokoro, H. Molecular Conformation and Packing of Poly(vinylidene fluoride). Stability of Three Crystalline Forms and the Effect of High Pressure. *Polym. J.* **1972**, *3*, 591.
(57) Lando, J. B.; Olf, H. G.; Peterlin, A. Nuclear Magnetic Resonance and X-Ray Determination of the Structure of Poly-(vinylidene fluoride). *J. Polym. Sci., Part A-1: Polym. Chem.* **1966**, *4*, 941−951.
(58) Gurnani, R.; Kamal, D.; Tran, H.; Sahu, H.; Scharm, K.; Ashraf, U.; Ramprasad, R.; polyG2G. A Novel Machine Learning Algorithm Applied to the Generative Design of Polymer Dielectrics. *Chem. Mater.* **2021**, *33*, 7008−7016.
(59) Wu, C.; Chen, L.; Deshmukh, A.; Kamal, D.; Li, Z.; Shetty, P.; Zhou, J.; Sahu, H.; Tran, H.; Sotzing, G.; Ramprasad, R.; Cao, Y. Dielectric Polymers Tolerant to Electric Field and Temperature Extremes: Integration of Phenomenology, Informatics, and Experimental Validation. *ACS Appl. Mater. Interfaces* **2021**, *13*, 53416−53424.
(60) Kamal, D.; Tran, H.; Kim, C.; Wang, Y.; Chen, L.; Cao, Y.; Joseph, V. R.; Ramprasad, R. Novel High Voltage Polymer Insulators Using Computational and Data-Driven Techniques. *J. Chem. Phys.* **2021**, *154*, 174906.
(61) Kamal, D.; Wang, Y.; Tran, H. D.; Chen, L.; Li, Z.; Wu, C.; Nasreen, S.; Cao, Y.; Ramprasad, R. Computable Bulk and Interfacial Electronic Structure Features as Proxies for Dielectric Breakdown of Polymers. *ACS Appl. Mater. Interfaces* **2020**, *12*, 37182−37187.
(62) Abbott, L. J.; Hart, K. E.; Colina, C. M. Polymatic: A Generalized Simulated Polymerization Algorithm for Amorphous Polymers. *Theor. Chem. Acc.* **2013**, *132*, 1334.
(63) Afzal, M. A. F.; Browning, A. R.; Goldberg, A.; Halls, M. D.; Gavartin, J. L.; Morisato, T.; Hughes, T. F.; Giesen, D. J.; Goose, J. E. High-Throughput Molecular Dynamics Simulations and Validation of Thermophysical Properties of Polymers for Various Applications. *ACS Appl. Polym. Mater.* **2021**, *3*, 620−630.
(64) Liao, L.; Huang, C.; Meng, C. Study on Mechanical Properties of Polyethylene with Chain Branching in Atomic Scale by Molecular Dynamics Simulation. *Mol. Simul.* **2018**, *44*, 1016−1024.
(65) Chen, L.; Wang, S.; Yu, Q.; Topham, P. D.; Chen, C.; Wang, L. A Comprehensive Review of Electrospinning Block Copolymers. *Soft Matter* **2019**, *15*, 2490−2510.
(66) Huang, J.; Turner, S. R. Recent Advances in Alternating Copolymers: The Synthesis, Modification, and Applications of Precision Polymers. *Polymer* **2017**, *116*, 572−586.
(67) Li, L.; Raghupathi, K.; Song, C.; Prasad, P.; Thayumanavan, S. Self-Assembly of Random Copolymers. *Chem. Commun.* **2014**, *50*, 13417−13432.
(68) Feng, C.; Li, Y.; Yang, D.; Hu, J.; Zhang, X.; Huang, X. Well-Defined Graft Copolymers: From Controlled Synthesis to Multipurpose Applications. *Chem. Soc. Rev.* **2011**, *40*, 1282−1295.
(69) Beginn, U. Gradient Copolymers. *Colloid Polym. Sci.* **2008**, *286*, 1465−1474.
(70) Che, S.; Fang, L. Porous Ladder Polymer Networks. *Chem.* **2020**, *6*, 2558−2590.
(71) Lee, J.; Kalin, A. J.; Yuan, T.; Al-Hashimi, M.; Fang, L. Fully Conjugated Ladder Polymers. *Chem. Sci.* **2017**, *8*, 2503−2521.
(72) Zhao, X.; Chen, X.; Yuk, H.; Lin, S.; Liu, X.; Parada, G. Soft Materials by Design: Unconventional Polymer Networks Give Extreme Properties. *Chem. Rev.* **2021**, *121*, 4309−4372.
(73) Xia, D.; Wang, P.; Ji, X.; Khashab, N. M.; Sessler, J. L.; Huang, F. Functional Supramolecular Polymeric Networks: The Marriage of Covalent Polymers and Macrocycle-Based Host-Guest Interactions. *Chem. Rev.* **2020**, *120*, 6070−6123.

https://doi.org/10.1021/acs.jctc.2c00022
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX