Liftover_2D: Fast Chromatin Interaction Coordinate Conversion (Open2C)

The implementation can be viewed at: https://github.com/heilcheng/liftover_2d

Applicant Profile

- **Name:** Hailey Cheng (Cheng Hei Lam)
- **Email:** heilcheng2-c@my.cityu.edu.hk
- **School/University:** City University of Hong Kong
- **Graduation Date:** 07/2027
- **Major/Focus:** Mathematics & Computer Science, with focus in Computational Biology
- **Country:** Hong Kong
- **Timezone:** UTC +8:00
- **GitHub profile:** https://github.com/heilcheng

GSoC INFORMATION

- **Previous GSoC participation:** No
- **Other organizations applied to:** DeepChem, for a small project.
- **Hours per week:** Taking 2 summer courses, but will be able to dedicate ~40 hours/week to the project.

Project information

1. Project Abstract

Liftover_2D will develop a fast and effective tool for converting chromatin interaction coordinates between different genome assemblies. As part of the Open2C ecosystem, it will enable researchers to "lift over" Hi-C and other 3C+ contact data without re-mapping raw reads, using existing chain files for coordinate translation. The tool will support both .pairs and .cool formats, using optimized algorithms that preserve pairwise context while achieving high performance through vectorization and parallelization, addressing key limitations of existing tools.

2. Detailed description

Liftover_2D addresses the challenge of converting chromatin interaction data (pairs of genomic loci) between genome assemblies. Preserving the pairwise nature of interactions is crucial, a capability often lacking in standard 1D genomic feature liftover tools. The tool will leverage chain files to map coordinates, using efficient data structures and algorithms to process millions of pairs quickly.

Core features:

- Fast conversion of coordinate pairs between assemblies.
- Native support for both .pairs and .cool file formats.
- Optimized interval-based lookups (aiming for O(log n) complexity per query).
- Robust handling and clear reporting of unmappable regions and ambiguities.
- Seamless integration with Open2C's existing toolchain (cooler, pairtools).

Comparison with Existing Tools (e.g., HiCLift):

While tools like HiCLift also perform liftover for Hi-C contacts using chain files, Liftover_2D aims to provide significant advantages:

- **Performance:** Optimized algorithms leveraging vectorized operations (NumPy), parallel processing (where applicable), and efficient interval tree implementations are designed for high throughput, aiming to significantly outperform existing solutions, especially on large, high-resolution datasets.
- **Direct .cool Support:** Liftover_2D will natively support the widely-used .cool format. This allows direct conversion of binned contact matrices without intermediate, potentially lossy, conversion steps (e.g., .cool -> .pairs -> lift -> .pairs -> .cool), preserving resolution-specific structures and simplifying workflows.
- **Ecosystem Integration:** As a core component of Open2C, it ensures seamless interoperability with cooler and pairtools for integrated analysis workflows, promoting standardization and ease of use within this established ecosystem.
- **Robustness & Transparency:** Specific attention will be paid to clear handling and detailed reporting of unmappable regions and potential mapping ambiguities, providing users with more control and insight into the conversion process.
- **Maintenance & Community:** Being part of Open2C ensures ongoing maintenance, community support, and alignment with evolving standards in the field.

Implementation details:

The core mapping relies on efficiently querying an interval tree built from the chain file.

```
# Conceptual core mapping function
def lift_coordinate(chain_tree, chrom, position):
    """Map a single coordinate from source to target assembly."""
    # Check if chromosome exists in the chain tree
    if chrom not in chain_tree:
        return None # Or appropriate handling for unmapped chromosome

    # Find overlapping chain segments in the interval tree
    overlaps = chain_tree[chrom].find(position, position + 1)
    if not overlaps:
        return None # Coordinate is in an unmapped gap

    # Select the best chain based on alignment score (heuristic for ambiguity)
    # Note: Real implementation might need more sophisticated ambiguity handling
```

```
    best_chain = max(overlaps, key=lambda x: x.data['score'])

    # Calculate the offset within the chain segment
    # Ensure correct handling of strand/direction if chains include it
    chain_offset = best_chain.data['target_start'] - best_chain.begin
    target_pos = position + chain_offset

    return (best_chain.data['target_chrom'], target_pos)
```

Algorithm Approach and Accuracy Assessment:
The core coordinate conversion algorithm relies on chain files. For a given coordinate within a mapped segment defined by a chain block, the conversion target_pos = source_pos + (target_start - source_start) is **deterministic**.
However, practical implementation involves choices and inherent limitations:

- **Chain File Limitations:** Chain files represent alignment scores and may not cover the entire genome or may contain ambiguities.
- **Ambiguous Mappings:** When a coordinate overlaps multiple chain blocks (e.g., near segment boundaries or in regions with paralogy), we initially plan to select the chain with the highest alignment score (max(overlaps, key=lambda x: x.data['score'])). This is a deterministic heuristic, but we will provide reporting on such instances. Alternative strategies might be explored based on mentor feedback.
- **Unmappable Regions:** Coordinates not covered by any chain block cannot be mapped and will be handled according to a user-defined policy (e.g., dropped with reporting, flagged).

Accuracy assessment will be multi-faceted:

- **Mapping Rate:** Report the percentage of input pairs successfully lifted to the target assembly.
- **Benchmark Validation:** Testing against known coordinate conversions (e.g., lifting between human genome builds hg19/hg38) and comparison with UCSC liftOver for single coordinates.
- **Consistency Check:** Where feasible, compare lifted contact maps (especially .cool matrices) to maps generated natively by re-mapping reads to the target assembly using metrics like Stratum-Adjusted Correlation Coefficient (SCC), Insulation Score correlation, and visual inspection of features (e.g., TADs, loops).
- **Simulation:** Use synthetic datasets with known relationships to verify correct recovery.
- **Reporting:** Provide clear statistics on unmapped pairs and pairs involving ambiguous coordinate mappings.

3. Significance and Motivation

Broader Importance:
Genome assemblies are fundamental references in genomics, yet they are constantly updated

and refined. Furthermore, comparing genome organization across different species is crucial for evolutionary studies. Liftover_2D addresses a critical bottleneck: the need to compare chromatin interaction data (like Hi-C) generated on different assemblies. Currently, the primary alternative is re-mapping raw sequencing reads to the new assembly, a computationally intensive and time-consuming process. This tool will:

- **Enable Comparative Genomics:** Facilitate studies of 3D genome evolution by allowing direct comparison of contact maps between species or assemblies.
- **Maximize Data Re-use:** Unlock the potential of vast public Hi-C datasets generated on older assemblies, making them usable with current references without costly re-processing.
- **Integrate Multi-Omics Data:** Allow researchers to consistently map and integrate 3C+ data with other genomic datasets (ChIP-seq, RNA-seq, variant calls) that might exist on different assembly versions.
- **Accelerate Research:** Provide a fast, accessible method for labs to keep their analyses up-to-date with the latest genome references or perform cross-assembly comparisons.

Personal Motivation:
My academic background merges Mathematics, Computer Science, and a strong interest in Biology, particularly computational biology (reinforced by my past Biology Olympiad experience). I am fascinated by the computational challenges inherent in extracting biological insights from large-scale genomic data. The 3D structure of the genome is a frontier in biology, and tools that help decipher it are essential.
I am motivated by the opportunity to:

- **Solve a Real-World Problem:** Address a practical need within the genomics research community.
- **Apply My Skills:** Leverage my experience in Python, algorithm design, and handling biological data to build an efficient and robust tool.
- **Contribute to Open Source:** Participate in the collaborative development of essential scientific software within the respected Open2C ecosystem.
- **Learn and Grow:** Deepen my understanding of 3D genomics, software development best practices, and performance optimization.

4. Future Directions and Broader Impact

Potential Future Enhancements:

- Support for lifting over annotations associated with contacts (e.g., loop calls, TAD boundaries).
- Handling more complex genomic rearrangements beyond those captured in standard chain files.
- Development of a comprehensive benchmarking suite.
- Exploration of alternative mapping strategies for ambiguous regions.
- Potential integration into larger bioinformatics workflow platforms.

Implications for the Research Community:
Liftover_2D has the potential to:

- **Become a Standard Tool:** Provide a reliable, efficient, and validated method for contact map liftover, improving reproducibility.
- **Save Resources:** Dramatically reduce the computational cost and time required for cross-assembly analyses compared to re-mapping.
- **Democratize Analysis:** Make sophisticated comparative 3D genomics accessible to a wider range of researchers.
- **Enable New Discoveries:** Facilitate large-scale studies comparing genome architecture across evolution, development, or disease states that are currently challenging.

5. Project Challenges and Approach

Which aspect of the project idea do you see as the most difficult?
The most challenging aspect will be optimizing for performance and memory efficiency at scale. Processing potentially billions of interactions (.pairs format) or large matrices (.cool format) requires:

- **Efficient Chain File Parsing:** Handling large chain files without excessive memory usage.
- **Optimized Interval Tree:** Ensuring the interval tree lookups are truly fast and scale well.
- **Batch Processing:** Minimizing Python overhead when processing millions or billions of coordinates.
- **.cool** Bin Mapping: Correctly and efficiently mapping data between potentially different binning schemes in the source and target assemblies.
- **Robust Edge Case Handling:** Systematically addressing unmappable regions, chromosome naming differences, and mapping ambiguities.

I plan to address these challenges by:

- Using optimized interval tree libraries or implementations.
- Implementing chunked/streaming processing for large files.
- Vectorizing coordinate mappings with NumPy where possible.
- Carefully designing the .cool bin mapping logic.
- Leveraging parallelization for independent tasks (e.g., processing chromosomes separately).

Which aspect of the project idea do you see as the easiest?
The conceptual approach to lifting over paired coordinates (map each end independently using chain files) is relatively straightforward. Implementing the basic command-line interface (CLI) structure and integrating with existing Open2C file I/O utilities (cooler, pairtools) should also be manageable standard tasks.

Which portion of the project idea will you start with?

I'll begin with the core single-coordinate mapping engine based on chain files. This involves:

1. Implementing an efficient chain file parser and interval tree builder.
2. Developing and testing the lift_coordinate function against known examples and tools like UCSC liftOver.
   This foundational piece is essential before building the pairwise (.pairs) or binned (.cool) logic on top.

6. Weekly timeline

- **Community Bonding:** Study Open2C codebase, set up environment, review formats, communicate with mentors, finalize strategy.
- **Week 1:** Implement chain parser, core single-coordinate mapping function, initial validation.
- **Week 2:** Implement pairwise mapping, batch processing, basic pairs I/O.
- **Week 3:** Develop pairs CLI, streaming/chunking, logging.
- **Week 4:** Handle edge cases (unmappable, chr names), optimize memory for pairs.
- **Week 5:** Begin cooler support, bin coordinate lifting design.
- **Week 6 (Midterm):** Initial cooler support, pixel mapping, validation tools, midterm report/demo. Ensure pairs conversion is functional.
- **Week 7:** Finalize cooler implementation, handle complex bin cases, metadata transfer.
- **Week 8:** Performance optimizations (parallelization, memory), benchmarking.
- **Week 9:** Validation testing (compare lifted vs. native, multiple assemblies), bug fixing.
- **Week 10:** Polish CLI/API, advanced features (reporting), refine error handling, ensure Open2C integration.
- **Week 11:** Write comprehensive documentation, examples, tutorials, prepare for code review.
- **Week 12:** Final testing, fix remaining issues, polish documentation, prepare final submission.

7. Other commitments

Taking 2 summer courses, but will be able to dedicate ~40 hours/week to the project.

8. Biography/About Me

Sophomore in Mathematics & Computer Science in CityUHK, based in Hong Kong, with a background in Biology (former Biology Olympiad student). Conducting research project on ODE-based models for drug combination prediction; implemented pathway extensions and multi-task neural networks under dept. of Mathematics. Built deep learning pipelines for

calcium imaging analysis using PyTorch for signal modeling and classification under dept. of Biomedical Engineering. Experienced in Python, NumPy, and handling biological datasets at scale. Eager to apply my interdisciplinary skills to develop impactful bioinformatics tools.