LinChemIn: Route Arithmetic — Operations on Digital Synthetic Routes

Marta Pasquini* and Marco Stenta

Syngenta Crop Protection AG, Schaffhauserstrasse, 4332, Stein, AG, Switzerland

E-mail: marta.pasquini@syngenta.com

Abstract

Computational tools are revolutionizing our understanding and prediction of chemical reactivity by combining traditional data analysis techniques with new predictive models. These tools extract additional value from the reaction data corpus, but to effectively convert this value into actionable knowledge, domain specialists need to interact easily with the computer-generated output. In this application note, we demonstrate the capabilities of the open-source Python toolkit LinChemIn, which simplifies the manipulation of reaction networks and provides advanced functionality for working with synthetic routes. LinChemIn ensures chemical consistency when merging, editing, mining, and analyzing reaction networks. Its flexible input interface can process routes from various sources, including predictive models and expert input. The toolkit also efficiently extracts individual routes from the combined synthetic tree, identifying alternative paths and reaction combinations. By reducing the operational barrier to accessing and analyzing synthetic routes from multiple sources, LinChemIn facilitates a constructive interplay between Artificial Intelligence and human expertise.
Introduction

The size of the corpus of chemical reaction data obtained from public, proprietary, and licensed sources has progressively grown over the years, matched by an increasing demand for extracting more value from chemical experiments. Scientists use computational tools and smart search methods to navigate this data effectively. For example, the Network of Organic Chemistry (NOC) approach converts individual chemical reactions into a graph-like object, allowing for graph-based searches and the discovery of new synthetic routes. Additionally, the application of predictive modeling has led to the development of Computer-Aided Synthesis Planning (CASP) tools, which provide actionable insight in the form of synthetic plans to molecular targets. To address the need for common frameworks across multiple CASP/NOC tools, including the manual input of experts, we extended the Python toolkit LinChemIn with new functionalities. By merging digital synthetic routes from multiple sources (both CASP platforms/models and input from experts), LinChemIn identifies novel connections between reactions, even ones not initially present in the initial inputs. This approach offers multiple alternative paths to strategic intermediates, enhancing the decision-making process and, ultimately, improving synthetic route design. This application note provides an overview of the concepts introduced in LinChemIn’s new module and outlines its usage. It showcases the route operations enabled by the toolkit through a practical case study. The note concludes with a glimpse of the future development roadmap. The source code of LinChemIn is freely available on GitHub at https://github.com/syngenta/linchemin and it is open to feedback and contributions from the community.

Methods

The data model implemented in LinChemIn (SynGraph) maps the first ontological layer of reaction networks (CASP output and NOCs) to directed graphs where REACTANT and
PRODUCT relationships connect Molecule and Chemical Equation nodes. LinChemIn allows the user to set the level of structural sensitivity while assigning unique identifiers to nodes by selecting among an extensible range of structure-derived molecular hashes such as the Canonical SMILES (capturing all structural information), non-isomeric Canonical SMILES (ignoring stereoisomeric information), standard and non-standard InChIKey (ignoring tautomeric information). Chemical Equation hash codes derive from Molecule hash codes through a simple combination logic inspired by the Reaction InChI and extended to other molecular hashes. Using a structure-derived molecular hash as a unique node identifier is a convenient expedient to control node and graph structural equality.

Among the possible sub-graph types of a reaction network, a small set of graph architectures bears special chemical meaning. The Synthetic Route is a particularly relevant example because it maps (with a set of Chemical Equation and Molecule nodes) the chemical synthetic steps necessary and sufficient to synthesize a target molecule (root node) from a set of starting materials (leaf nodes). By dissecting the Synthetic Route, we define the Synthetic Path as a linear sequence of Chemical Equation and Molecule nodes linking the root node (synthetic target) and one leaf node (starting material); we can extend the concept to include linear paths connecting two nodes within the NOC. The combination (union) of multiple Synthetic Routes sharing a common root gives a Synthetic Tree. Moving one level up, the combination (union) of multiple Synthetic Trees, each stemming from a different root, leads to a Synthetic Forest, one of the (potentially many) connected sub-graphs contained in a chemical reaction network like the NOC. Linked by a clear hierarchical relationship, these distinct graph architectures provide a basic description of the reaction network (Figure 1, panel a).
Figure 1: a) Examples of the elementary synthetic graph architectures described in the text. b) merge/mine operations on graph architectures.

**Route Operations**

**Single route operations**

**Route identification**  A *SynGraph* instance is a value object entirely defined by its properties: the set of unique nodes and their relationships. *SynGraph* instances are identified as *Synthetic Routes* if they match some completeness criteria defined at the chemistry (presence of all necessary and sufficient synthetic steps to produce a chemical target) and graph (unique root, correct node, and relationship labels) levels.

**Route editing**  Any change to either or both the set of nodes and relationships encoded into a *SynGraph* leads to a distinct *SynGraph* instance. Users can edit *Synthetic Routes* by adding or removing *Chemical Reaction* nodes from the graph while ensuring the chemical consistency of the output synthetic route. This functionality enables users to modify routes predicted by CASP tools by removing undesired steps or by adding necessary ones based
on other predictions, literature data, direct experience, or chemical intuition. The common operational framework ensures that the properties (descriptor, metrics, etc.) of the new route can be compared with those of the original one, thus enabling property-driven interactive route design.

Multi-route operations

Route Equality SynGraph instances are value objects because they are entirely defined by their constituting nodes, object values themselves, whose identity stems from the chemical structure they represent. This approach leverages the underlying Python data model to simplify the equality assessment between pairs of Synthetic Routes.

Merging Merging two or more Synthetic Routes into a Synthetic Tree corresponds to the union operation between the individual set of nodes (see Figure 1, panel b)). To this aim, a node equality assessment enables a meaningful combination of multiple routes, removing redundant nodes and keeping edges appropriately to keep original node-relationship information.

Mining The mining (or extracting) Synthetic Routes from a Synthetic Tree is the inverse operation of merging. This operation is linked to the “route enumeration ” from NOC\textsuperscript{35} and, in our case, is tailored to operating on a Synthetic Tree. To ensure a chemically sensible route reconstruction, the mining procedure relies on a modified depth-first search algorithm that leverages the directional nature of the Synthetic Tree and responds to the local node connectivity. Starting from the root node (target) the algorithm moves toward the leaf nodes (starting materials), memorizing in a stack the nodes discovered until a divergence point (node “OR ”, see Figure 1, b) is encountered: a Molecule node connected through PRODUCT relationships with multiple Chemical Equation nodes. This situation represents, from a synthetic standpoint, a chemical that multiple reactions can produce and hence identify alternative routes to the target. The algorithm identifies these bifurcation
points by dynamically analyzing node connectivity and, after duplicating the node stack, keeps accumulating nodes along each branch defined by these Chemical Equation nodes. The exhaustive exploration of the Synthetic Tree up to the leaves nodes yields a set of node stacks, each converted into a distinct Synthetic Route.

**Results**

To exemplify the new functionalities described above, we predicted synthetic routes for the antiviral drug Amenamevir.\textsuperscript{36} To highlight the versatility of the LinChemIn input interface, we harvested predictions from tree CASP platforms: IBM RXN\textsuperscript{19} (model trained with the NextMove Pistachio data-set), AstraZeneca’s AiZynthFinder\textsuperscript{17} (model trained with the NextMove USPTO data-set) and Askcos from MIT\textsuperscript{37,38} (model trained with the REAXYS data-set). It is worth noticing that, not aiming at a benchmark between platforms and models, we did not attempt to align training sets of prediction options. In the following examples, the canonical SMILES is the molecular hash that defines the identity of Molecules nodes; reagents are omitted from the chemical reaction hash (‘\texttt{r-p}’: reactants and products), ensuring that only reactant and products contribute to the identity of Chemical Equation nodes.

As a first step, we load the prediction output for the common target as saved by the individual CASP platforms. After reading the first six routes from each file, a dedicated LinChemIn façade yields a monopartite (reaction only) SynGraph instance for each input route. A preliminary duplicate check follows data intake to identify potential duplicate routes in the input set.

Listing 1: The output from the CASP tools are read and the routes converted into SynGraph objects.

```python
import json
from linchemin.interfaces.facade import facade

input_dict = {
    'data/IBMRXN_routes.json': 'ibm_retro',
    'data/AZ_routes.json': 'az_retro',
```

https://doi.org/10.26434/chemrxiv-2023-g84vw ORCID: https://orcid.org/0000-0001-6761-7142 Content not peer-reviewed by ChemRxiv. License: CC BY 4.0
At this stage, we use atom-to-atom mapping\textsuperscript{39} to challenge the role of reaction components and, if necessary, redistribute the individual chemicals among the reactant, reagent, and product roles. After ensuring reagents are excluded by the reaction hash, the workflow reconstructs routes, checks again for duplicates, and yields the unique routes. In this case, the initial set contained two routes differing only reagents in one or more reactions (see SI, routes ibm\textsubscript{1}, Figure SI-7, and ibm\textsubscript{2}, Figure SI-8), hence considered equivalent. This leaves only seventeen routes.

Listing 2: The role reassignment procedure based on the atom-to-atom mapping is performed.

```python
from linchemin.cheminfo.atom_mapping import perform_atom_mapping
from linchemin.cgu.syngraph_operations import extract_reactions_from_syngraph
from linchemin.cgu.syngraph import MonopartiteReacSynGraph

def atom_mapping(route_list: list, mapper: str) -> list:
    """ Performs the atom mapping of a list of chemical equations smiles """
    mapped_routes: list = []
    for route in route_list:
        reaction_list = extract_reactions_from_syngraph(route)
        out = perform_atom_mapping(reaction_list, mapper_name=mapper)
        syngraph = MonopartiteReacSynGraph(out.mapped_reactions)
        mapped_routes.append(syngraph)
    return mapped_routes or None

mapped_routes = atom_mapping(unique_initial_routes, 'rxnmapper')
unique_routes = []
for route in mapped_routes:
    if route not in unique_routes:
        unique_routes.append(route)
```
As a further check, we seek routes that are a subset of each other. This step removes one route more (see SI, route az\_5, showed in Figure SI-17, is a subset of route az\_4, showed in Figure SI-16), leaving sixteen unique routes.

**Listing 3: Subsets are removed.**

```python
subsets = facade("subsets", unique_routes)
for item in subsets:
    route_to_remove = next((r for r in unique_routes if r.uid == item[0]), None)
    if route_to_remove and route_to_remove in unique_routes:
        unique_routes.remove(route_to_remove)
```

Editing routes by adding or removing chemical steps is a key requirement for any informatics system that aims at leveraging the knowledge and experience of scientists alongside the predictive power of reactivity models. For this reason, besides the possibility of entering whole human-provided routes alongside CASP predictions, we ensure scientists can add or remove individual reaction steps from synthetic routes. These features improve the overall route design efficiency: rather than starting from a blank canvas, the scientists edit, if necessary, the route background already laid out by CASP tools. The example highlights the simplicity of both edit operations applied to the askcos\_3 route. (Figure 2).

**Listing 4: A route is edited by removing a chemical reaction and then by adding a new chemical reaction**

```python
from linchemin.cgu.syngraph_operations import remove_reaction_from_syngraph, add_reaction_to_syngraph

route = unique_routes[11]
node_to_remove = (f'\[CH3:17\][C:16]([CH3:18])([CH3:19])[O:14][C:13]=(O:15)\''
                  + f'\[NH:12\][C:3]1=[CH:2][CH:1]=[C:6]([CH:5]=[CH:4]1)[C:9],''
                  + f'1=[N:10][O:11][CH:7]=[N:8]1>>[NH2:12][C:3]1=[CH:2][CH:1],''
new_route_removal = remove_reaction_from_syngraph(route, node_to_remove,
                                                  remove_dandling_nodes=True)

node_to_add = (f'\[CC:0:3\][C:2]([O:1])=[CH:4]1[CH2:5][CH2:6][S:7]=[O:8]=[O:9],''
               + f'\[CH2:10\][CH2:11]=([O:1])=[C:2]([CH:3])=CH:41[CH2:5][CH2:6][S:7],''
               + f'=[O:8]=[O:9][CH2:10][CH2:11]1''
new_route_addition = add_reaction_to_syngraph(route, node_to_add)
```

The route mining example shows that the number of routes extracted from a Synthetic
Figure 2: a) the original route. b) the same route after the removal of a node. c) the same route with the addition of a new node.

Tree might be considerably larger than the number of routes used to build it. The merging operation creates in the resulting *Synthetic Tree* new valid cross-route synthetic pathways, possibly including multiple new divergence points ("OR" nodes). In our case, the procedure yields 150 individual routes after merging and mining from the initial 16. The procedure enriches the information contained in the original input, creating innovative combinations that differ in both size (number of steps) and complexity (number of branches)(Figure 3).

Listing 5: Synthetic Routes are mined from the Synthetic Tree obtained by merging the original predictions.

```python
from linchemin.cgu.route_mining import mine_routes

root = 'Cc1cccc(C)c1N(CC(=O)Nc1ccc(-c2ncon2)cc1)C(=O)C1CCS(=O)(=O)CC1'
mined_routes = mine_routes(unique_routes, root)
```

https://doi.org/10.26434/chemrxiv-2023-g84vw ORCID: https://orcid.org/0000-0001-6761-7142 Content not peer-reviewed by ChemRxiv. License: CC BY 4.0
Figure 3: Frequency of a) number of steps, i.e., number of Chemical Equations and b) number of branches for the original set of routes and the mined routes.

Conclusions

In this contribution, we highlighted the newly developed functionalities of LinChemIn, an open-source Python toolkit for editing and analyzing synthetic routes. These functionalities aim to bridge the gap between digital resources (data, models, predictions) in synthetic chemistry and end-users, enabling. Through a case study, we showcased the key features offered by LinChemIn: the identification of unique routes, their chemical-aware manipulation, the possibility of merging routes into a synthetic tree, and mining synthetic routes from a tree. This work is a step toward a data-driven and model-enabled route design and selection. We aimed at lowering the barrier for a scientist to interact with model output, taking inspiration from them and editing whenever necessary or appropriate. A streamlined program interface will support software developers in using the toolkit to develop custom workflows. In the future releases of the toolkit, we will enhance the analytical capabilities of the toolkit to yield quantitative assessment of or route properties through the calculation of descriptors and metrics.
Competing Interests

The authors declare that they have no competing interests.

Author’s Contributions

Marco Stenta: Conceptualization, Methodology, Supervision, Writing - Original Draft, Writing - Review & Editing. Marta Pasquini: Methodology, Software, Validation, Writing - Original Draft, Writing - Review & Editing.

Supporting Information Available

Pictures of the synthetic routes used in the case study are reported in the Supplementary Information file.

A Jupyter Notebook, alongside the data to reproduce the work described in the article, is available at:

https://github.com/syngenta/LinChemIn_publications/LinChemIn_RouteArithmetic

All the examples require linchemin version 2.2.5 and linchemin_services version 1.0.0.

The source code of LinChemIn is freely available on GitHub at

https://github.com/syngenta/linchemin

The source code of linchemin_services, containing services accessible via API are available at https://github.com/syngenta/linchemin_services

References


(32) Dhaked, D. K.; Ihlenfeldt, W.-D.; Patel, H.; Delannée, V.; Nicklaus, M. C. Toward a Comprehensive Treatment of Tautomerism in Chemoinformatics Including in


