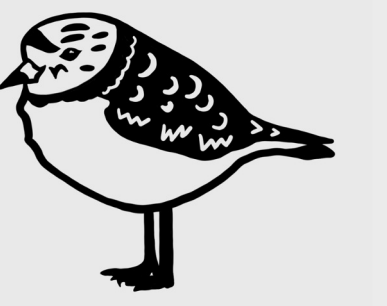


PLOVER DB: A SPEEDY, SPECIALIZED GRAPH DATABASE

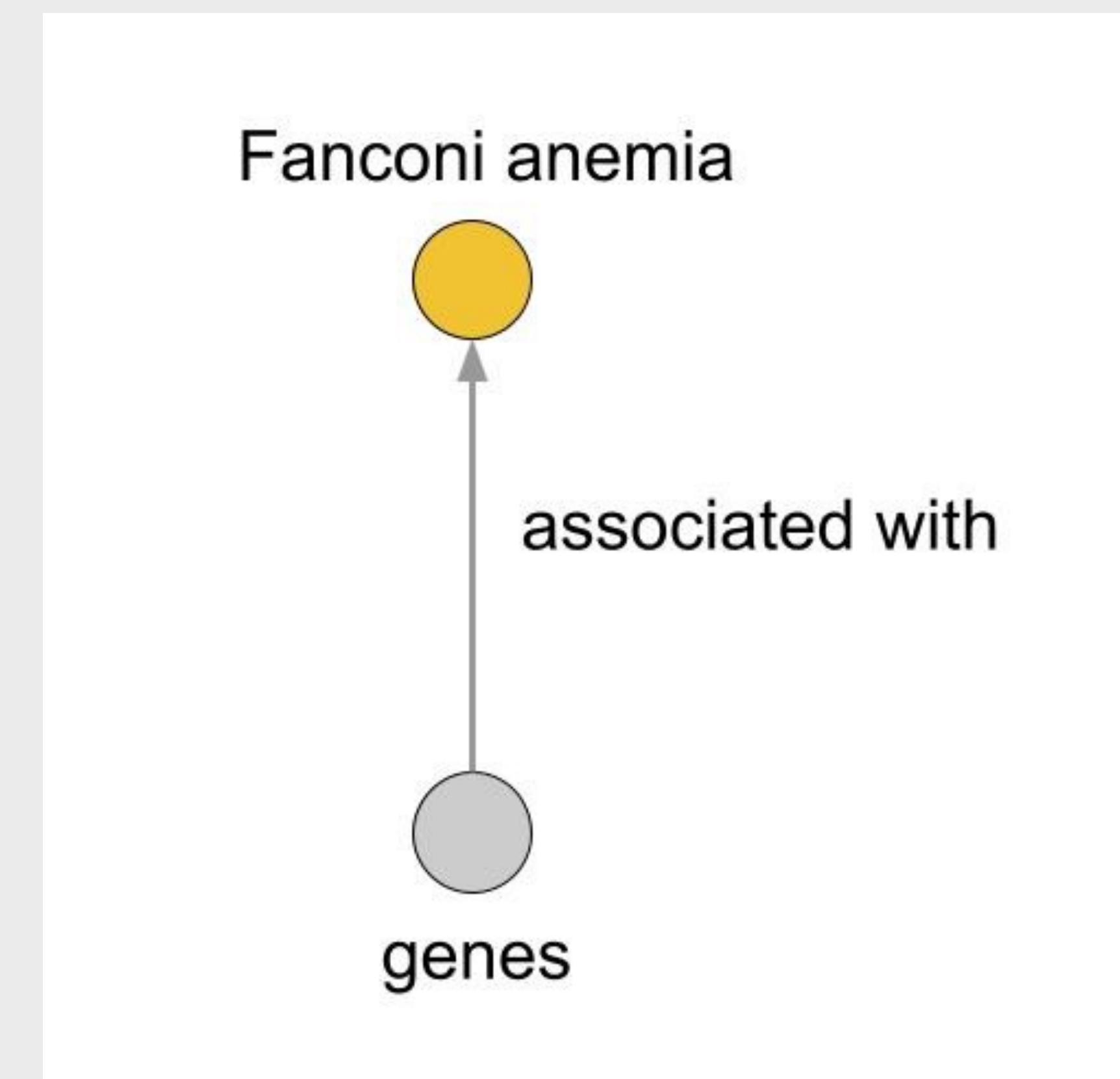


By Amy Glen (Advisor: Stephen Ramsey, PhD)

INTRODUCTION

- RTX-KG2 is a large network (or “graph”) of biomedical knowledge integrated from over 70 different sources, developed by the Ramsey Lab [1]
- It includes ~6 million biomedical concepts and ~40 million relationships between those concepts
- For instance, it contains information on which drugs treat which diseases, what the protein targets of different drugs are, which diseases are associated with mutations in which genes, etc.
- As part of the larger NCATS Biomedical Data Translator project [2], RTX-KG2 needs to be able to speedily answer queries
- Those queries have a very specific structure, which we call “one-hop” queries, because they are asking for one kind of relationship; see Figure 1 for an example
- We originally used Neo4j [3], a popular graph database platform, to store RTX-KG2 data behind the scenes and produce answers to any queries
- Neo4j has a wide variety of functionalities and is a useful platform in many ways, but was prohibitively slow for many of the queries submitted to RTX-KG2
- Because the kinds of queries sent to RTX-KG2 have such a specific structure, we hypothesized that we could build our own graph database platform from scratch that is specifically designed for answering such queries, and thus is more performant
- We named that database platform PloverDB

FIGURE 1: EXAMPLE QUERY



METHODS

PloverDB is written in the Python programming language. We designed it to be:

- In-memory: All of its data is stored in RAM, rather than on the hard drive; this allows it to answer queries more quickly
- Read-only: RTX-KG2 only receives queries asking it to *retrieve* information, never to *add* information to the database, so Plover only supports “read” vs. “write” queries, which greatly simplifies its design
- Containerized: Plover runs in Docker, which means it can be run on any operating system

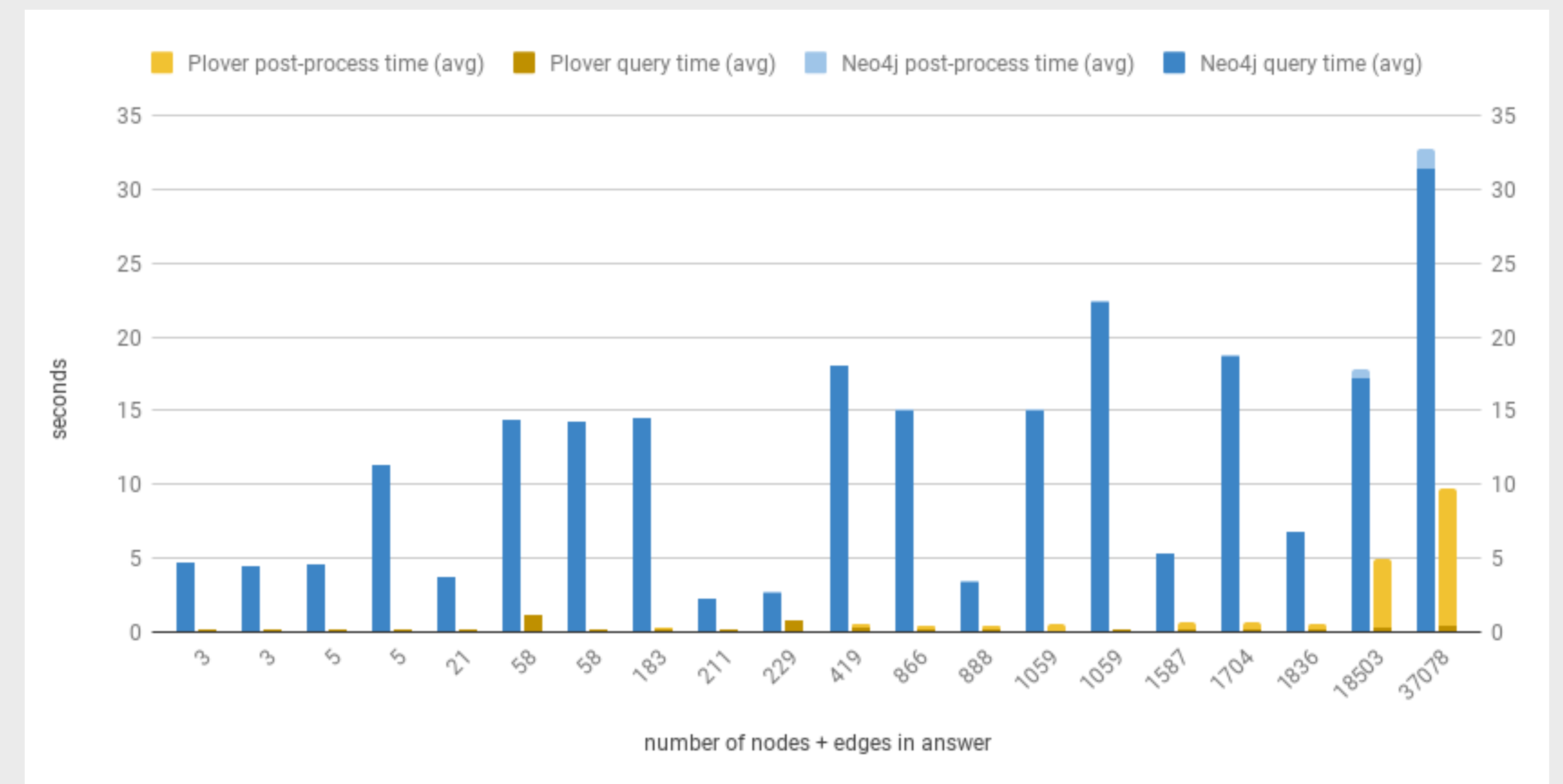
To efficiently answer one-hop queries, Plover has highly-tailored nested indexes (which can be thought of as look-up maps that allow for very fast retrieval of concepts by a key).

Plover is accessible for querying at <https://kg2cploverdb.transltr.io> and its source code is available on GitHub (github.com/RTXteam/PloverDB).

CITATIONS

1. Wood, E.C., Glen, A.K., Kvarfordt, L.G. et al. RTX-KG2: a system for building a semantically standardized knowledge graph for translational biomedicine. *BMC Bioinformatics* 23, 400 (2022). doi:10.1186/s12859-022-04932-3
2. Fecho K, Thessen AE, Baranzini SE, et al. Progress toward a universal biomedical data translator. *Clin Transl Sci.* 2022 May 25;15(8):1838–47. doi: 10.1111/cts.13301.
3. Neo4j: Graphs for Everyone. <https://github.com/neo4j/neo4j>

FIGURE 2: QUERY RUNTIMES WHEN USING NEO4J VS. PLOVER DB



RESULTS

- Plover requires about 100 GB of RAM to run (when loaded with RTX-KG2 data)
- We found that for a set of typical queries, Plover took 1% - 30% of the time that Neo4j did to return the answer (Fig. 2)
- As queries get larger, the advantage of Plover appears to decrease, though it still offered an ~3x speed improvement over Neo4j for the largest query tested
- For larger queries, a greater portion of Plover’s runtime goes to post-processing, which essentially includes decorating the returned biomedical concepts and relationships with information that is useful for a human, but isn’t required to answer the query (e.g., text descriptions of concepts, publications supporting a given relationship, etc.)

CONCLUSIONS & DISCUSSION

- PloverDB is an in-memory, read-only graph database platform that is tailored for answering one-hop biomedical queries
- We found that it answers typical RTX-KG2 queries in 8% of the time required by Neo4j on average
- Due to this, we adopted PloverDB as the data-hosting platform behind our RTX-KG2 knowledge graph
- Future work should include 1) assessing how much using a “split” system, as Plover does, where “decorative” data is stored locally on the machine hosting the RTX-KG2 API improves the Neo4j times and 2) comparing to further alternative solutions (other graph database platforms, Datalog, etc.)

