**1.1 Describe (a) the problem to be solved or addressed by the invention and (b) any known solutions and drawbacks to those existing solutions.** Please cite any relevant technical documents. Documents can be uploaded via link below. For software related inventions, describe whether there is a technical problem (e.g., in the field of computers or technological arts) that is being solved or addressed by the invention.

In the past, the rules for recommending alternative drugs were stored in slow and costly relational databases. Our new invention stores all the data needed to execute millions of business rules in a high-performance knowledge graph. Because we leverage a highly simplified in-RAM pointer traversal algorithm, our system has both higher performance and lower operational costs.

**1.2  Describe the invention, including how it solves the problem described in Section 1.1.** Describe how the invention works, in a manner that someone having the requisite technical experience could implement or practice the invention. For software-related inventions, describe how the invention provides a technical solution to a technical problem (e.g., in the field of computers or technological arts). If possible, include all essential structural/system components and process steps. Include or attach any relevant flowcharts, drawings, technical specifications, presentations, and other supporting documents.

|  |
| --- |
| The invention works by:   1. Modeling both pharmacy benefit plans, drugs, and rules in terms of a labeled-property graph which is composed of both vertices and edges. This is stored in a VertexType and EdgeType file similar to a traditional RDBMS data description language. This step was done in conjunction with drug formulary rule subject-matter experts. 2. Creating a Subgraph that contains the VertexTyes and EdgeTypes. This was done using a deployment script system written in Python (PyTigerGraph). 3. Adding existing plan, drug, and rule data from legacy systems into the subgraph database. We tested this using an existing database of approximately 2.3 million rules imported from the legacy RxBuilder tools. 4. Creating a short (100-line) graph query that traverses this graph to arrive at an alternative drug recommendation. 5. This graph query is then executed on a distributed graph engine that can leverage thousands of concurrent threads of execution.   The net result is that the rules run quickly, efficiently and the system scales to be able to serve millions of rule execution per minute. |