Practical Application of Graph Data for Agent-based Models

Clarence Dillon[1][0000−0002−4739−0558]

George Mason University, Fairfax, VA 22030, USA cdillon2@gmu.edu
https://cos.gmu.edu/cds/

Abstract. The schema-free and object-oriented nature of graph databases make them useful companions to agent-based modelling and simulation. This is an overview of a use case where a Neo4j graph database is used to support a simulation of world order. It reviews steps to recreate a database that includes examples of how it is used to support initialization of the simulation with empirical facts and validation of simulation results.

Keywords: Graph Database · Validation · Agent-Based Model.

1 Introduction

Agent-based models (ABM) simulate change in a system over time. Validation of ABM typically focuses on comparing empirical evidence to the resulting state of a simulation, patterns of change while the simulation runs, or both. Graph database schema are inherently object-oriented, which parallels the programming model in ABM. Here I describe some design choices and present some practical examples of an application that uses a graph database with an ABM that simulates the social processes that generate the world's political order. I imported historical data about world order into a Neo4j[1] graph database in order to initialize a simulation and store the simulation's output to support analysis and validation. The resulting database represents spatial, temporal, and event-based facts. Practical experiences creating and using this graph database in conjunction with ABM may be useful, not only to the academic community studying international relations and peace science but, also to those searching for ways to manage empirical and simulation data in their research.

The database project has also solved some technical issues for initializing ABMs that are grounded with published datasets that, in most cases, were organized to support statistical modeling in environments such as STATA[2], SPSS[3], or R[4]. Statistical and mathematical models differ from ABM in that the former are collected (either unnormalized or in a normal form) to be easily subitized for analysis. ABM require data to denormalized–disassembled and attributed to programmatic objects–and then organized into a chronological order by discrete time steps calibrated to simulation steps. The bulk of peace science datasets are organized first around nation states (individually or by dyad) and

time (by year of record or event date). Examples include the volume of trade between two states during a given year or the beginning and end of a war between two states. Using the trade example: in an agent-based simulation, each state agent needs to track its own exports to or imports from each other state agent on every simulation step. Graph databases, such as Neo4j, have some advantages over relational databases in cases when data elements are highly related and the schema needs to be flexible. This report highlights a few of these advantages by presenting examples of the graph representations of imported data as well as examples of how data thus organized can be queried to support a simulation.

2 Methods

This section reviews the data models and methods to integrate the temporal, spatial and peace science components of the database. The three components can be recreated following three steps described here. The most significant of these–step 2–is the integration of several peace science datasets imported as graph data, with limited prior editing (where possible). The import scripts (over 12,000 lines of code, divided into 13 files) are available for review (and code contributions) in the author's online code repository [5]. The datasets can be downloaded from their own, original online repositories (listed in the References and the code repository). Neo4j[1] makes community and enterprise versions of the database software available for download from their website; using the Desktop application is recommended for development and testing, however. The nodes and relationships of the database are represented as objects in the simulation code and use an object-graphical model (OGM) interface to load them into the simulation and save them to the database.

The first step in generating the database is to provide a temporal context for the spatial component and the imported peace science data. A calendar tree is a graphical depiction of years, months and days, including their inter-relations. Since version 3.4, Neo4j has had a `date` data type, but having a calendar tree can still be helpful. In this case, time steps for the simulation of world order are calibrated to weeks and a calendar tree is added to relate dates to the weeks of the year, following a model depicted in figure 1. This mapping of weeks to years follows the International Standards Organization (ISO) 8601 date specification[6]. Periodic data is related to the year of record while event data is related to the week from which it began and until the week in which it concluded. The simulation code defines *Week* and *Year* classes, which map directly to database nodes labelled as *Weeks* and *Years*.

The second step is to import historical data about world order sourced primarily from the Correlates of War (COW) project, but includes some others. These data include: the international system [7], conflict (wars and disputes) [**?**,**?**], trade [10], alliances [**?**], national capabilities [12], religion [13], diplomatic exchange [14], membership in international organizations [15], and facts about each polity [16]. Maintaining source metadata for each data element while attributing it to the relevant entities is accomplished by creating nodes represent-

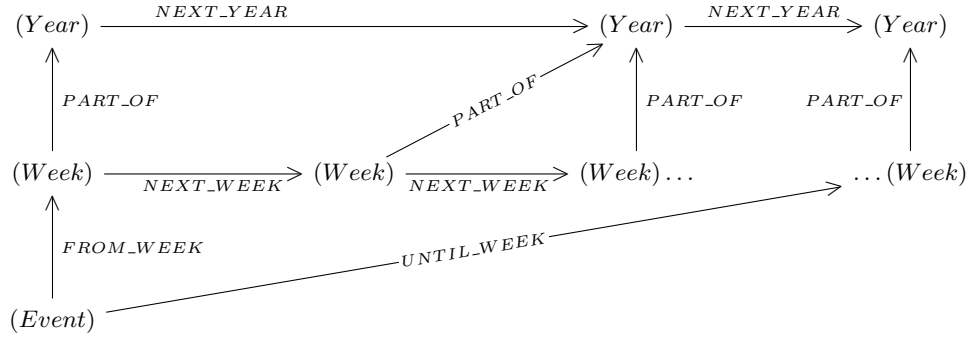

Fig. 1. Event dates mapped to a calendar tree.

ing sources of the data and the datasets they provide. *Sources* and *Datasets* are the provenance of each *Fact* they contribute. Rather than importing these data as simple relationships, such as depicted in figure 2, we divide the relationship and insert the *Fact*, as depicted in figure 3. Note: the text references *Facts* nodes generically, but in the database they are double labelled with the type of facts they represent, *e.g. Membership Fact* in figure 3. These graphical data models simultaneously represent an international relations ontology within the simulation, as *States* and *Systems*

Fig. 2. Relationship without Facts.

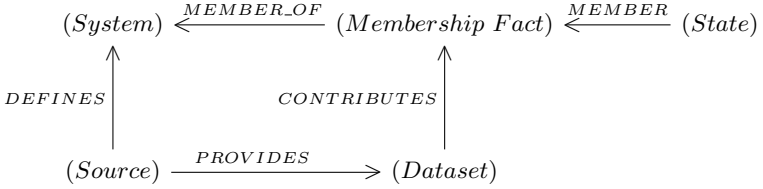

Fig. 3. State system membership with Fact and metadata

Managing source metadata for the imported data as well as the simulation data is required to fully denormalize the database and keep from mixing meta-

data with *Facts* in the data. The *Dataset* and the *Source* nodes are labelled with metadata, such as the extent of the data, the version, publication date, author(s), *etc.*; such as is defined in the Dublin Core Metadata Elements [20]. The simulation is itself represented in the database as a data source and each model run, uniquely identified by the current time in milliseconds, is a dataset. Facts generated in the course of a simulation are connected to the dataset, allowing each simulated fact to be identified and attributed to the simulation run that produced it.

The third step is significant because it manifests an ontological design consideration for the simulation which distinguishes between states and the territories they occupy. The spatial data also does not include changes to territories over time, but represents snapshots of the world map during certain years: 1816, 1880, 1914, 1938, 1945 and 1994. The database models *Territories* structurally as proxies for *Facts* which were valid during the year of the snapshot. The territorial data is available for review [17], however, unlike the peace science data and the temporal data the source of *Territories* cannot be validated. A computation subdivides the approximate boundaries of *Territories* into hexagonal *Tiles*. In practice, *Territories* are a low-resolution collection of hexagonal *Tiles* with artificially geometrical boundaries. Figure 4 depicts the data model for *States*, *Territories* and *Tiles* where *States* occupy *Territories* and *Territories* include *Tiles*.

The computation which generates the hexagons reads a GeoJSON feature collection file which has been edited by the author to include COW country codes, where appropriate. Tiles are calculated from an icosahedral Snyder equal area discrete global grid [18] using the H3 hierarchical indexing system in resolution 4 hexagons [19]. Each resolution 4 hexagon bounds an area of approximately 1,700 km^2 or about 23 km per side. At this resolution, there are 288,122 tiles (288,110 hexagons and 12 pentagons) to cover the surface of the Earth, however, only 81,428 hexagons are required to cover the land territories and coastlines. Only those are included in the database, as no aspect of the simulation project requires simulating activity in ocean areas. The resolution is fine enough that the area of large territories represented as hexagons is fairly accurate, but just tolerable for small countries. For example, Luxembourg is the smallest nation state included in the COW system in 1816 at about 2,500 km^2, represented as a single tile.

Fig. 4. NMC data attributes population to states

Representing territories as tiles is necessary for the world order simulation, where each tile represents human, economic and natural resources; all agents in the ABM. The graph database schema allows data state-level peace science data to be easily distributed and attributed to tile-level agents that interact in the simulation. For example, the National Military Capabilities (NMC) [12] data provides historical populations for states in the COW system. The graph data representation follows the model in figure 5. A configuration setting in the simulation of world order calls a method which assigns a population value to each tile following a Zipf distribution (see figure 6) so that the total population of the territory–hence the state–is approximately equal to the NMC value.

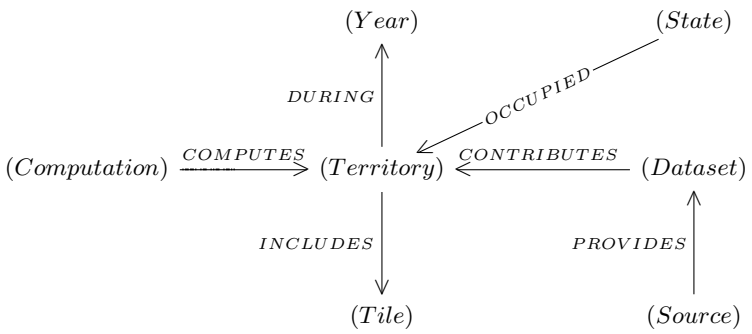

Fig. 5. Distinguishing between states and the territory they occupy

3 Results

The results of the imports and computations is a graph database that contains approximately 5 million nodes and 20 million relationships. Together, these entities hold nearly 27 million property values. Though many relationships have no properties and most properties belong to nodes. Stored on disk, the dataset represents 2.19 gigabytes (GiB). The final dataset combines the calendar tree, the several peace science datasets, and territories comprised of hex-tiles; all related such that the database can be queried by entity, event, time and space.

Figure 7 represents a simplified meta-graph of the node labels and relationship types, revealing high-connectedness of some nodes in the data. The node and relationship labels are much too small to read but highlights in red help to orient the reader in resulting metagraph. Importantly, the most central and most connected node represents *States*. There are also highly connected nodes on the far left representing *Datasets* and to the far right representing *Years* during which events occurred or for which data is recorded. The nodes arranged in columns immediately to the right of the States node represent (for the most

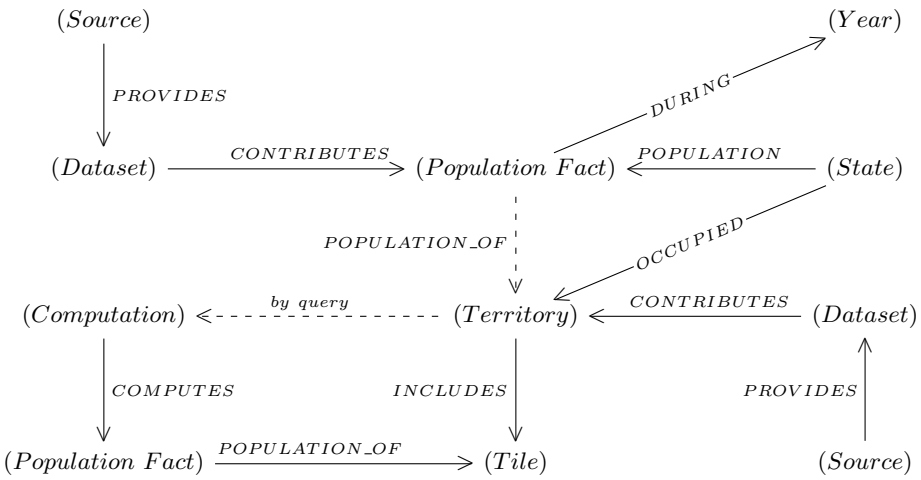

Fig. 6. Applying population data through territories to tiles

part) various *Facts*: War Participation facts, System Membership facts, Territory facts, Trade facts and facts from the National Military Capabilities [12] dataset, among others. To the right of Facts nodes are nodes that represent other system entities: *Wars*, *Alliances* (generally and by type), *Inter-Governmental Organizations*, *etc.* It is evident in the bottom-half of the figure that many *Facts* about *States* relate primarily to the *Years* for which they are recorded. In the top-half of the figure, *Facts* about *States* are related both to these other system entities and to the *Weeks* or *Years* in which the relationships existed.

The graph can be queried for simple relationships as well as deep relationships and analytic computation. Four useful examples follow, highlighting specific types of queries.

The first example is simple query that asks for *States* that became members of a *System* between 1816 and 1825, or where the membership date is null.

Listing 1.1. Simple relationship query.

```
MATCH (s:State)-[m]->(mf:MembershipFact)-[mo]->(y:System)
WHERE 1816 <= mf.from.year <=1825 OR mf.from IS NULL
RETURN s, mf, y
```

In this case, the database browser can be used to return results as a visualization of the relationships between *States*, *Facts*, and *Systems*. Because the query does not specify which system is of interest, the database returns results for the COW State System and for the Major Powers–a sub-system which is *PART_OF* the State System. Nodes representing *States* are common to both systems but the *Facts* which assert system membership uniquely mediate each relationship.

The second example returns a frequency table that can be used directly within a statistical computing environment. In this example, a deep query returns all of the weeks since 1 January 1815 in which a *State* initiated either a

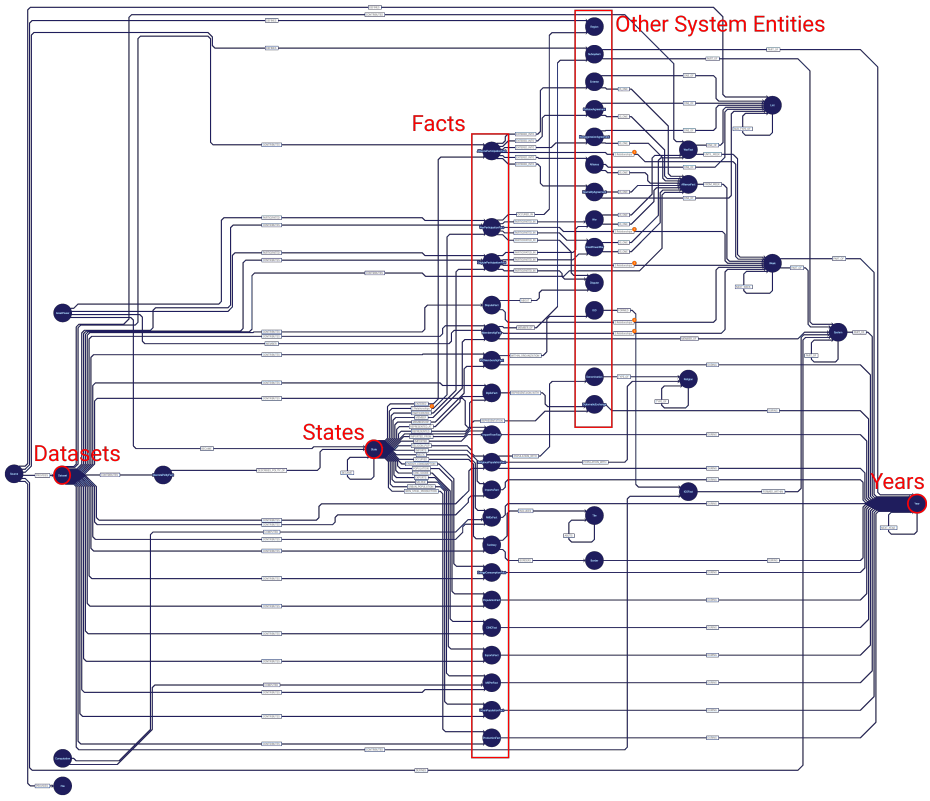

Fig. 7. Simplified meta-graph depicting the node labels and relationship types for the combined dataset.

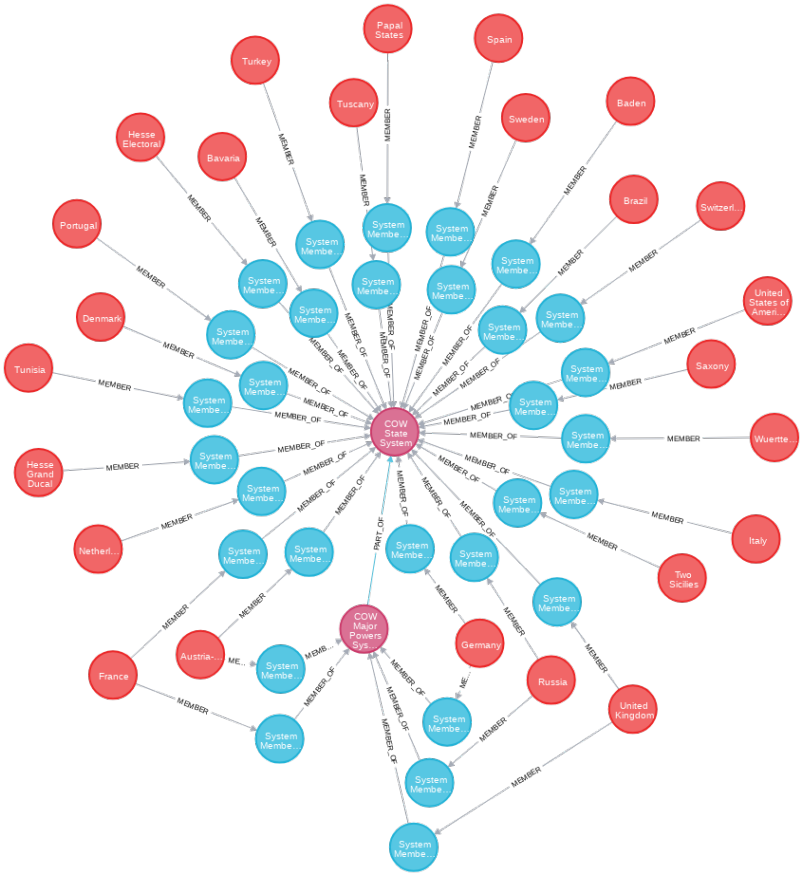

Fig. 8. Example with direct relationships and filtering.

war or interstate dispute. The table includes the week number a war or dispute was initiated, the number of such events that were initiated that week, and the number of states that initiated those events. This query was used in the simulation to calculate the frequency with which the simulation initiates conflicts between states in the system.

Listing 1.2. Deep query example

```
MATCH (fw:Week)-[:FROM_WEEK]-(f:WarParticipationFact)-[:PARTICIPATED_IN]
  -(w:War), (f)-[p:PARTICIPATED{initiated:true}]-(s:State)
WITH collect({week:fw.stepNumber, wars:w, part:f, init:p}) AS rows
MATCH (fw:Week)-[:FROM_WEEK]-(f:DisputeParticipationFact{
  originatedDistpute:true})-[:PARTICIPATED_IN]-(d:Dispute),
  (f)-[p:PARTICIPATED]-(s:State)
WITH rows + collect({week:fw.stepNumber, wars:d, part:f, init:p}) AS allRows
UNWIND allRows AS r
WITH r.week as week, r.wars as wars, r.part as part, r.init as init
RETURN DISTINCT week,
  count(DISTINCT wars) as wars,
  count(part) as states ORDER BY week
```

Table 1. Conflict frequency in weeks since 1 Jan 1815 and the number of states that initiated those conflicts.

week	81	140	166	...	402	431	...
wars	1	1	1		1	1	
states	2	2	2		2	1	

Example three is a deep query that includes spatial as well as temporal dimensions. In a simulation, it was intended for each polity to consider which territorial neighbors and neighbors of neighbors were potential targets of military conquest while excluding partners in alliances other than peace entente. Agreements need be entered into prior to 1816 and not concluded before then.

Listing 1.3. Spatial and temporal dimensions in queries.

```
MATCH (t:Territory{mapKey:"Prussia of 1816"})-[:OCCUPIED]-(p1:Polity)
  -[e:ENTERED]-(apf:AlliancePartipationFact)-[:ENTERED_INTO]-(a:Alliance)
  -[:ONE_OF]-(l:List), (a)-[:ENTERED_INTO]-()-[e2:ENTERED]-(o:Polity)
WHERE e.from.year <= 1816 AND e.until.year > 1816 AND l.type <> "Entente" AND
  e2.from.year <= 1816 AND e2.until.year > 1816
WITH COLLECT(o) AS allies, t
MATCH (t)-[:BORDERS{during:1816}]->(:Border)-[:BORDERS{during:1816}]
  -(n:Territory)-[:BORDERS{during:1816}]-(:Border)-[:BORDERS{during:1816}]
  -(o:Territory)
WHERE t <> n AND t <> o
WITH COLLECT(n) + COLLECT(o) AS ter, t, allies
UNWIND ter AS z
MATCH (z)-[:OCCUPIED]-(p:Polity)
WHERE NOT p IN allies
WITH COLLECT(p) as potential
RETURN potential
```

The result for Prussia in 1816 includes 13 states: Denmark, France, Italy, the Kingdom of the Two Sicilies, the Netherlands, the Papal States, Portugal, Spain, Sweden, Switzerland, Turkey, Tuscany, and the United States of America.

The final example calculates the betweenness centrality of territories in the 1816 map. The query relies on two database applications that are provided with the Neo4j Desktop application: Graph Algorithms and Awesome Procedures on Cypher (APOC). It warrants mention because it is an example of how these applications extend the capabilities of the database and the utility of the dataset. The Graph Algorithms application can calculate the betweenness centrality of nodes in a sub-graph a single type of relationship. However, the data model specifies *Border* nodes between *Territories*.

Listing 1.4. Territories and Borders data model.

```
(Territory)-[:BORDERS]->(Border)<-[:BORDERS]-(Territory)
```

The APOC application allows us to create "virtual nodes" or "virtual relationships" that exist as proxies within single query, even though the data does not explicitly exist in the dataset. Here it is used to specify a virtual relationship between *Territories* via *Borders* nodes and *:BORDERS* relations so that the Graph Algorithms calculation of betweenness centrality can use the simplified relationship to transit the sub-graph.

Listing 1.5. Query example with graph algorithm predicated on virtual relationship.

```
CALL algo.betweenness.stream(
  'MATCH (t:Territory{year:1816}) RETURN id(t) AS id',
  'MATCH (w:Territory)-[:BORDERS{during:1816}]-(b:Border)<-
    [:BORDERS{during:1816}]-(t:Territory)
    WHERE w<>t AND w.name <> "World Oceans" and t.name <> "World Oceans"
    WITH t, w, apoc.create.vRelationship(w,"NEIGHBORS", {during:b.year}, t)
    AS rel RETURN id(w) AS source, id(t) AS target',
  {graph:'cypher', write:false,  direction:'both', relationship:'rel'}
)
YIELD nodeId, centrality
MATCH (t:Territory) WHERE id(t) = nodeId
RETURN t.name AS territory, centrality ORDER BY centrality DESC
```

Table 2. Betweenness centrality of five most central 1816 map territories.

territory	centrality
Russian Empire	3791.9522
Unclaimed 5	3721.2833
Ottoman Empire	3181.6423
Egypt	2616.6833
Prussia	1863.4437
⋮	

4 Discussion

The examples in section 3, are intended to provide a glimpse of how data in a graph database can be extracted for use in modeling and simulation. This

section provides additional information about each of the examples in 3, above and points out the significance of those examples. This project to collect and integrate peace science data into a graph began as an effort to support one particular research project, but has proven useful for other analysis. The general approach used in this project may benefit other research areas with multiple, related datasets.

4.1 Discussion of Findings

The first query example at listing 1.1 is a straight forward database filter and join operation and is a typical application of relational databases. It is included here to orient the reader to the Cypher query language (CQL) and point out that the basic capabilities of relational databases are available in graph databases, as well. It also demonstrates some features of the CQL syntax. In this case, the query acknowledges that we expect there to be some relationship between *States* and *MembershipFacts* as well as between *MembersipFacts* and *Systems*, but it need not be specified. The CQL 'MATCH' statement is similar to a standard query language (SQL) 'SELECT' statement. The 'WHERE' and 'RETURN' clauses are similar in both languages. If there were multiple sources of *MembershipFacts*, such as Gleditsch's "Extended State System Data" [**?**] then it would be necessary to specify which sources or datasets must be related to the *Facts*.

The example in listing 1.2 combines data from two sources: a calendar tree and the COW Inter-state War Data [8]. The query searches for deeper relationships than in the first example, in that it looks first for weeks during which *States* participate in armed conflict and then relates those *ParticipationFacts* to conflicts and to the *States* that initiated those conflicts. This query also demonstrates aggregation (counts) and the 'DISTINCT' key word, which simplifies further interpretation of the data, making it easier to consume in the simulation program. Following the mediated facts data model, these results could also be filtered by the source of the data in cases when there are competing datasets or multiple versions of data available. Again, this feature is not provided in the example.

The third query, shown in listing 1.3 is significant, not only because it combines states with other system entities (alliances) but also with temporal and spatial dimensions of the data. It is also a good example of when a graph database is more useful than a relation database. This query is selective, in that the first part of the query returns a single result (the territory occupied by Prussia in 1816). From there, it includes eight relationships–which would have required three temporary tables and 12 joins in a relational database. The graph database query is not only more simple to conceptualize and write, but likely executes faster; taking only 13 milliseconds over a network connection.

Listing 1.5 for the final example demonstrates the utility of employing virtual data to represent implied relationships and uses that virtual data to calculate node properties on a sub-graph, using a common graph centrality metric. The Neo4j software used in this project includes seven different procedures to measure centrality, six for community detection, seven path-finding algorithms, five

similarity measures, and seven link prediction algorithms. Advanced users can write their own procedures if the included methods do not satisfy their specific analytic requirements. Using these methods is relatively easy, once the query has been correctly formulated. The query is only eight lines long in the example, but within a simulation (where the results need not be printed out for review) is only four lines of code.

4.2 Discussion of Broader Implications

The graph database used here was developed to support initialization, verification and validation of a particular agent-based simulation but it has wider utility. The database can be accessed with connectors for many programming languages and programming environments; Java and Python being the foremost languages used in ABM toolkits such as MASON, REPAST, or MESA and Java being the language underlying popular environments such as NetLogo and Gama where models are specified with higher-level scripts.

The database can be accessed via statistical programming languages such as Python (again) and R. This allows, for example, a script used in the statistical examination of an empirical dataset to be repeated directly on data collected from a simulation; thus supporting validation and study of simulation data. In another use case, missing values from empirical data can be imputed and saved back to the database. By using a model where:

Listing 1.6. Managing provenance of computed values in a dataset.

```
(:Dataset)-[:CONTRIBUTES]->(:Fact)<-[:COMPUTED]-(:Computation)
```

It is possible to use these computed facts within simulations and computations when desired, and exclude them when necessary.

4.3 Implications for future research

The simulation research that this database supports is still developing and some of the implementation details of the data project will change to meet the evolving requirements of the simulation. It is expected that the simulation of wars, peace agreements, trade volume, *etc.* of each simulation run will be saved to the database with simulation run details stored as the provenance of that data. Stored queries, such as those presented in this report, will be used to compare simulated institutional artifacts to the historical institutions represented in the data.

The data included in the dataset used in this project is limited, the spatial data has not been validated and a great deal of historical non-state data has not been included. Future work might expand on the available data and add competing data. For example, Patrick Manning [22] has published data on historical populations in African territories, which could supplant data not included in the National Military Capabilities data, which includes only states.

There also exists an opportunity to verify published datasets when the same fact can be asserted by more than one source. The data model used here can

further be supplemented with the sources of facts in each published dataset, allowing more robust criticism, which is a necessary process in good science.

Mentioned in section 2, the published datasets integrated into this database are imported by script that require little or no editing of the data before importing. This is an important feature of this project as well as any other projects that implement this method. Down-stream projects such as this should place no expectations on the authors who publish the source data. Importing data by script also supports validation of the import methods and validation of the graph data representation.

5 Conclusion

Graph databases are robust and flexible solutions to storing data that was not designed to support ABM. Denormalizing structured datasets into a graph schema adapts data to the object-orientation and relatedness that characterize ABM, without sacrificing analytic requirements of scientific analysis. The schema-free nature of graph data is a natural environment to integrate separately-published datasets into a single repository that can be adapted to a variety of uses and querying. Certainly, graph data has made it possible to integrate data from algorithmic, computed and static sources. Maintaining accurate metadata and the provenance of each fact within the database has proven to be a critical component of the database and is required to support validation. This use case–initialization and validation of world order simulation–is open to further development. By providing the simulation code and import scripts, this example can help similar efforts to integrate data in other knowledge domains.

References

1. Neo4j. https://neo4j.com/. Last accessed 10 May 2019.
2. Stata Software. https://www.stata.com/. Last accessed 10 May 2019.
3. IBM SPSS Software. https://www.ibm.com/analytics/spss-statistics-software. Last accessed 10 May 2019.
4. R Core Team (2019). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/.
5. Dillon, Clarence W. "World Order Data". Online, https://github.com/usuallycwdillon/vvna. Last accessed 15 May 2019.
6. International Standards Organization "Date and Times Format". Online, https://www.iso.org/iso-8601-date-and-time-format.html. Last accessed 10 May 2019.
7. Correlates of War Project (2017). "State System Membership List, v2016." Online, http://correlatesofwar.org. Last accessed 10 May 2019.
8. Sarkees, Meredith Reid and Frank Wayman (2010). "Resort to War: 1816 - 2007". Washington DC: CQ Press.
9. Palmer, Glenn, Vito D'Orazio, Michael R. Kenwick, and Roseanne W. McManus. Forthcoming. "Updating the Militarized Interstate Dispute Data: A Response to Gibler, Miller, and Little". International Studies Quarterly.

10. Barbieri, Katherine and Omar M. G. Omar Keshk (2016). "Correlates of War Project Trade Data Set Codebook", Version 4.0. Online, http://correlatesofwar.org. Last accessed 10 May 2019.
11. Gibler, Douglas M. (2009). International military alliances, 1648-2008. CQ Press.
12. Singer, J. David. (1987). "Reconstructing the Correlates of War Dataset on Material Capabilities of States, 1816-1985" International Interactions, 14: 115-32.
13. Zeev Maoz and Errol A. Henderson (2013). "The World Religion Dataset, 1945-2010: Logic, Estimates, and Trends." International Interactions, 39: 265-291.
14. Bayer, Reat (2006). "Diplomatic Exchange Data set, v2006.1." Online: http://correlatesofwar.org. Last accessed 10 May 2019.
15. Pevehouse, Jon C., Timothy Nordstrom, and Kevin Warnke (2004). "The COW-2 International Organizations Dataset Version 2.0," Conflict Management and Peace Science 21:101-119.
16. Polity IV: Regime Authority Characteristics and Transition Datasets. Online, http://www.systemicpeace.org/inscrdata.html. Last accessed 10 May 2019.
17. Dillon, Clarence W. "Historical Base-maps". Online, https://github.com/usuallycwdillon/historical-basemaps. Last accessed 15 May 2019.
18. Sahr, Kevin, Denis White, and A. Jon Kimerling (2003). "Geodesic Discrete Global Grid Systems". Cartography and Geographic Information Science, No. 2, 30:121-134.
19. Uber. "H3: A hexagonal hierarchical geospatial indexing system" Online, https://uber.github.io/h3//. Last accessed 15 May 2019.
20. International Standards Organization Dublin Core Metadata Elements. Online https://www.iso.org/standard/71339.html. Last accessed 10 May 2019.
21. Gleditsch, Kristian Skrede (2004). "A Revised List of Wars Between and Within Independent States, 1816-2002" International Interactions 30: 231-262
22. Manning, P. (2014). African Population, 16502000: Comparisons and Implications of New Estimates. In E. Akyeampong, R. Bates, N. Nunn, J. Robinson (Eds.), Africa's Development in Historical Perspective (pp. 131-150). Cambridge: Cambridge University Press. doi:10.1017/CBO9781139644594.006