# Neo4j4Us Proposal

A system using graph databases and Bayesian Networks to assist with a student's planned curriculum.

Edwin Kassier
Department of Computer Science
University of Cape Town
Cape Town, South Africa
KSSRUB001@myuct.ac.za

Josh Redelinghuys
Department of Computer Science
University of Cape Town
Cape Town, South Africa
RDLJOS002@myuct.ac.za

## 1. PROJECT DESCRIPTION

Many students struggle to complete their Computer Science (CS) degree in the standard three-year period, leading to drop-outs and course repeats. The department is interested in developing a system to address this problem by creating a Machine Learning (ML) system built on top of a Neo4j database with the potential to analyse student academic performance and provide feedback on a student's planned curriculum. The proposed system will use the courses a student plans to enrol in as input to generate a report. This report will indicate the likelihood of a student completing their degree in 3 years (as determined by the trained Bayesian Net (BN)), as well as feedback on whether the proposed curriculum violates any requirements of their degree. Other features will include the indication of past student performance among those who completed the proposed courses (determined using queries on a Neo4j graph database).

## 2. PROBLEM STATEMENT & PROJECT SIGNIFICANCE

Student attrition in the UCT Department of CS is a major cause for concern, seeing that the IT sector in South Africa is currently experiencing a deficit in the number of skilled graduates entering the field [1]. High attrition rates also negatively affect the reputation of the Department of CS, as well as having a negative impact on student morale. To increase the likelihood of academic success, student morale can be improved by ensuring that they enrol in the courses most suited to them. If a student is advised that they may not cope with their planned curriculum, they may plan to change their field of study. This results in reduced student drop-out and course repeats as well as reducing potential student debt incurred.

## 3. RESEARCH AIMS

The aim of the Neo4j4Us system is to create a unified platform that UCT's CS students may use as an aide when designing their academic curriculum. The principal goal of this system is to notify students of potential problems they may encounter while pursuing their planned degree, as well as their likelihood to succeed with an acceptable degree of accuracy. This system is to be trained using the anonymised transcript and matric result data from UCT CS students between 2012 and 2018. The resultant outcome of this system would be an increased throughput rate of UCT CS students, while also curbing drop-out.

### 3.1 Edwin - The Prediction Component

To explore how accurately a BN can predict the probability of student failure (a yes/ no answer to a particular student finishing their degree in the accepted time frame) amongst the UCT CS student population using an anonymised dataset of past student matric results and academic transcript data as training data, with exploration into which inputs are most useful in making these predictions. Depending on time constraints, this exploration can be extended to include predicting the chance of students qualifying for entrance into the CS honours degree. Additionally, an aim is to investigate this implementation utilising a graph database (Neo4j), as well as exploring how the accuracy of such a system can be improved through more efficient inference algorithms and native Neo4j machine learning functionalities.

### 3.2 Josh - The Advisory Component

To explore if a graph database (Neo4j) is an appropriate means of checking the constraints of a structured degree and further, to explore which procedure of determining similar students (using student data stored graphically) will produce the most accurate indications of students' performance in their degrees. The efficacy of this component may be measured subjectively, through the complexity involved in developing and using such a system, and objectively through comparisons with a relational database implementation of the same system.

## 4. PROCEDURES AND METHODS

The advisor system's proposed methods will be presented first, followed by the predictor's proposed methods.

### 4.1 A Neo4j Graph Database

The graph database to be implemented in Neo4j shall contain two graphs which serve two purposes; firstly, a constraint-checking tree is required to facilitate the checking of degree constraints. Secondly, a graph containing student data (including matric results, courses taken, and the marks achieved in those courses) will be used to group similar students and provide indications of the performance of past students enrolled in the same classes.

Once the simple constraint-checking graph representing the basic requirements of a degree has been constructed, and a straightforward query to report on the performance of similar students has been implemented, more detailed constraints can be incorporated, and experiments can be conducted.

#### *4.1.1 Check the Basic Constraints of a Degree Plan*

The constraints of a computer science degree will be represented in a tree. This tree will contain a start node to read in the planned curriculum of a student, question nodes which check various constraints of the degree and answer nodes which will classify the students plan as suitable or unsuitable for a computer science degree. The data input to this tree will be a list of courses a student has planned to enrol in. This data will be classified into an answer node (thereby indicating any errors with the planned curriculum) by traversing through question nodes such as "Do these courses constitute enough credits for the first year of studies?" connected by "Yes" and "No" edges.

The constraint-checking tree should be arranged in such a way that the number of question nodes required to reach an answer node is minimised. The most critical requirements (such as compulsory courses) of the degree must be checked first to identify any errors before the more specific requirements (such as courses which may not be taken together) are checked. Experimentation with the order in which constraints are checked will produce the most efficient and compact graph to traverse.

To design this graph, all the constraints specified in the UCT Science Handbook will be collected, formalised and represented in a low-fidelity graph (hand-sketched). Optimisations can then be made on the graph before being implemented in Neo4j. Then, using sample planned curriculums as input, the time taken to reach a decision will be used to determine the optimal arrangement of nodes.

To test this tree, a series of planned curriculums (some of which violate at least one of the constraints of the degree and some of which adhere to all requirements) will be input and checked against the correct result. Once the basic tree is complete and correct, further, more specific constraints (such as timetabling clashes) will be incorporated.

### *4.1.2 Report the Performance of Similar Students*

Student and academic data will be stored in nodes used for similarity queries. Such queries will be designed to find students who are similar to one another. Data retrieved from these similar students will be used to calculate the average course mark achieved (accompanied by standard deviation) in each of the planned courses.

A script is required to read each line of the current, cleaned relational database (which stores student data) into Neo4j. It must correctly transport data from tables to student and course nodes, whilst generating meaningful relationships between them.

Simply, the design of the graph will contain a student node connected to course nodes via edges such as `ENROLLED-IN` (with the obtained mark as an attribute). Initially, groups of similar students will be found by querying the student graph to find students with similar GPA's, matric results and those who are enrolled in the same courses. This allows for queries to determine the number of students enrolled in a course, the average mark obtained by enrolled students and the fail rate of the course.

This procedure will be the basis of providing feedback on how similar past students performed in their computer science degree. This database will also provide the backend data required for training the machine learning aspect of the project.

### *4.1.3 Experimentation and Comparison with an SQL Implementation*

To establish the suitability of Neo4j for this system, subjective and objective comparisons will be made against an SQL implementation. Subjectively, the ease-of-development and the complexity of the two systems will be compared to determine how capable Neo4j is of addressing the problem mentioned in Section 2. Objectively, the performance (execution time) of the queries on the student transcript data will be compared to discover any significant performance advantages of Neo4j over a relational table. Queries to find groups of similar students will be executed on both systems multiple times. The time taken for these queries to return the node groups will be recorded. A T-test will be used to determine any statistically significant difference in the performance of the Neo4j and SQL databases. These comparisons will provide evidence to the viability of graph systems for this system.

### *4.1.4 Experiment with Different Methods of Grouping Similar Students*

An investigation into what variables constitute 'similar' students will be conducted. This will be explored by first examining coarse-grain metrics such as similar GPA and a similar number of first year and senior courses enrolled in. Then, other metrics such as specific senior courses taken will be considered. Experimenting with these metrics should lead to more accurate advice.

Additional exploration of which query procedures (such as simple-count, Euclidean distance and cosine procedures as outlined in Building Recommendation Engines [3] and Section 6.3) produce the most accurate feedback on how similar students have fared in the past will be conducted.

## 4.2 Bayesian Net & Machine learning

Bayesian networks perform their prediction using an acyclic graph and Bayes Theorem (P[A|B] = P[B|A] / P[A]P[B]) to create and update probability tables that store the likelihood of the possible values that a node will have, based on the nodes related to it in the graph and their values. A BN will be created to perform supervised learning in the form of identifying students unlikely to finish their degree in the specified time frame with their academic history, having trained this system using the given student transcript data and matric performance of CS students between 2012 and 2018.

### *4.2.1 Integration of Neo4j with the Bayesian Network*

Neo4j offers multiple drivers for Java and Python, indicating that it is unlikely that there will be any issues in accessing and utilizing the graph data for a BN. Therefore, the investigation will rather focus on the performance differences of training times of the BN using each of the database types described in Section 4.1.3.

### *4.2.2 Training Bayesian Network to Achieve the Desired Accuracy*

As mentioned in the ML literature review (Section 6.1) for this project, there are many possible algorithms that could be used for training the network, commonly known as inference. From the literature review, the most promising candidate for training was shown to be the max-min hill climbing algorithm. It provided a slightly higher level of predictive accuracy than its contemporaries of the time, in addition to its more detailed documentation, making it easier for implementation into the proposed system.

### *4.2.3 Outputting the Rationale of the Bayesian Net*

Due to the nature of the project, it is essential for the system to output not only the likelihood of the student completing the degree in the allocated time but also the reasoning behind the probability in the case of at-risk students. The BN structure and the probabilities learnt during training, will provide this information.

### 4.2.4 *Testing the System Against a Currently Available Bayesian Net Tool*

There are caveats to open-source BN systems as the free versions often impose limits on the amount of data permissible for uploading to construct a BN. This renders these tools as unsuitable for testing against the proposed system that will train with the full dataset. In addition, open source systems are often so outdated that their use could skew results or would require an inordinate amount of alterations to suit the comparison needs of the project. This is attributed to the fact that many of them are enhanced BNs containing features beyond the scope of this project. A suitable free BN tool will be selected depending on the size of the data.

## 5. ETHICAL, PROFESSIONAL AND LEGAL ISSUES

As this research does not include participation from external parties, no ethical clearance regarding user-studies is required. The primary ethical consideration, however, involves the anonymity of the user data used as the basis of this project.

Student data including courses enrolled in and the results obtained in those courses must all be totally anonymised before use to respect the confidentiality of the users' university transcripts. Gender, sexuality and race will be excluded from the data used to avoid stereotype bias, so the personal information (gender, race, sexuality) of students are not required. Because only transcript data are used, the only ethical implication is that despite anonymization, it may still be possible to determine the identity of a student. This does not affect this project, however, as the results of queries on the database will not be publicly available - the data are used solely to provide predictions. End-users have no access to the database to examine the anonymised data.

The intellectual property (in the form of machine learning algorithms and databases) developed at the termination of this project shall remain the sole property of UCT and may be used for future research and development. The knowledge gained from development and experimentation (presented in the final report) shall be publicly available.

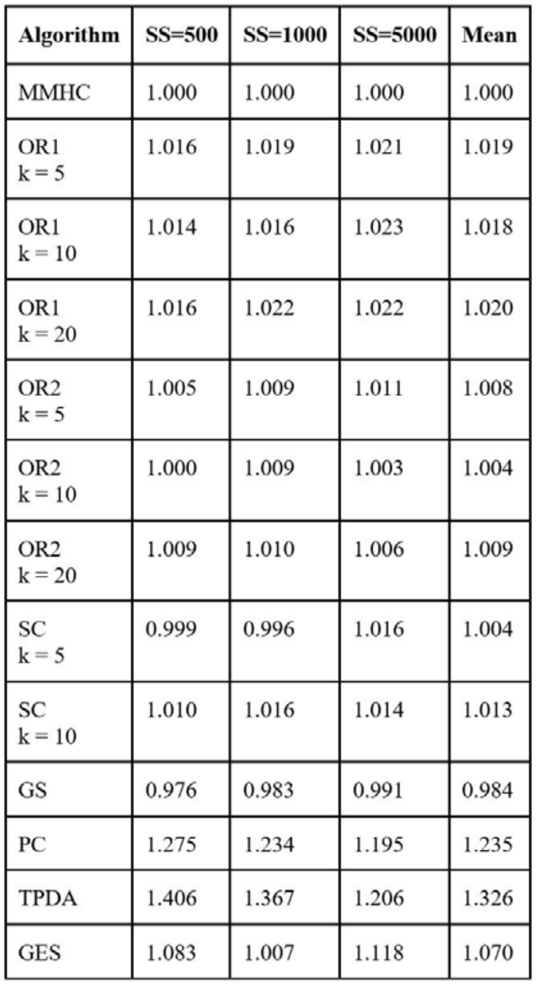
## 6. RELATED WORK

## 6.1 Bayesian Networks for Risk Classification

Inference algorithms define the methods that Bayesian Networks use to define the relationships between their explanatory variables. An inference algorithm will allow a Bayesian Network to identify the relationships between variables in its training process, allowing the algorithm to update the probability tables of its explanatory variables with respect to these relationships.

There are many types of inference algorithms. In a study performed in 2006, some of the most effective inference methods of the time were measured against a newly invented Max-Min Hill-Climbing (MMHC) algorithm for BN learning. The study compared the performance of the algorithms with training sets of 500, 1000 and 5000 values for each algorithm, averaging results over five tests [6]. The following table summarises the results for Bayesian score tests for a set of inference algorithms.

**Table 1: Summarised comparison of Bayesian network learning methods normalised relative to the MMHC method for Bayesian score results** [6]



| Algorithm | SS=500 | SS=1000 | SS=5000 | Mean |
|---|---|---|---|---|
| MMHC | 1.000 | 1.000 | 1.000 | 1.000 |
| OR1 k = 5 | 1.016 | 1.019 | 1.021 | 1.019 |
| OR1 k = 10 | 1.014 | 1.016 | 1.023 | 1.018 |
| OR1 k = 20 | 1.016 | 1.022 | 1.022 | 1.020 |
| OR2 k = 5 | 1.005 | 1.009 | 1.011 | 1.008 |
| OR2 k = 10 | 1.000 | 1.009 | 1.003 | 1.004 |
| OR2 k = 20 | 1.009 | 1.010 | 1.006 | 1.009 |
| SC k = 5 | 0.999 | 0.996 | 1.016 | 1.004 |
| SC k = 10 | 1.010 | 1.016 | 1.014 | 1.013 |
| GS | 0.976 | 0.983 | 0.991 | 0.984 |
| PC | 1.275 | 1.234 | 1.195 | 1.235 |
| TPDA | 1.406 | 1.367 | 1.206 | 1.326 |
| GES | 1.083 | 1.007 | 1.118 | 1.070 |

K = Maximum allowed size for the candidate parents' sets

Inference algorithms tested:

1. Sparse Candidate (SC), (Friedman, Nachman & Pe'er, 1999)
2. PC (Spirtes, Glymour & Scheines, 2000)
3. Three Phase Dependency Analysis (TPDA), (Cheng et al., 2002)
4. Optimal Reinsertion (OR) (Moore & Wong, 2003)
5. Greedy Hill-Climbing Search (GS, using the BDeu scoring)
6. Greedy Equivalent Search (GES), (Chickering, 2002)
7. Max-Min Hill-Climbing (MMHC, novel hybrid algorithm, (Brown, Tsamardinos & Aliferis, 2004)

The results of the Bayesian score tests, being a measure of prediction efficacy across multiple Bayesian networks, show that on average, the MMHC algorithm performed at the same, if not better, level as the Greedy Search, Optimal Reinsertions (types 1 and 2) and Sparse Candidate algorithms, with the other constraint-based algorithms performing worse. While these results are in favour of the MMHC algorithm, it must be noted that the authors of the study invented the algorithm and were thus likely biased in the types of tests performed to show the algorithm in a more positive light. While this research is relatively out of date, in terms of the rate of improvement in computer science, it is useful in analysing the various mechanisms used for inference and which types of mechanisms have been proven to be effective.

### 6.2 Neo4j Constraint-Checking Trees

Neo4j describes the procedure required to implement a constraint-checking tree on its platform as follows [5]. As these trees are more complex, Java methods which make use of the Neo4j Traversal API are required. The following components are required [2]:

- The Tree itself. Rule nodes will contain expression attributes (the condition applied in that rule e.g. age >= 21). Category nodes will have an ID and description. Rule nodes will be connected to other nodes by “Is True” and “Is False” relationships.
- The Evaluator. This method decides to stop traversing if an answer node has been reached or to continue down an edge determined by the expander.
- The Expander. This determines where to traverse by using the expression evaluator.
- The Expression Evaluator. This evaluates a rule based on the expression attribute stored in rule nodes, using the data input. The result of the evaluation determines the next edge to follow.

The graph is continually evaluated using a DFS until the data is classified in an answer node.

### 6.3 Neo4j for Recommendations

Historical data can be used to indicate past trends which influence recommendations. In the context of this project, providing an indication of how students in similar situations fared in the past is comparable to extracting information in the same way a recommendation engine produces suggestions.

Gorakala [3] outlines the process of building a recommendation engine in Neo4j. First, find related nodes which are either mutually connected to a node or have similar outgoing/ ingoing edges. Then, by using a count of the number of similarities between the nodes as a metric of their relatedness, extract data to recommend from the most related nodes.

Other methods (as outlined by Neo4j [4] and Building Recommendation engines [3]) can be used to determine the similarity between two nodes. The Euclidean distance between two nodes can be calculated to provide an indication of how related the two nodes are. A Euclidean distance of 0 indicates that two nodes are identical. This can be implemented formulaically (Equation 1) where p and q are the attributes of edges connecting two (student) nodes to a central (course) node. Alternatively, Cypher provides the `algo.similarity.euclideanDistance` method to calculate this similarity [4].

**Equation 1: Euclidean Distance**

$$d(\mathbf{p},\mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2}.$$

Another method, the Cosine similarity algorithm, determines the similarity of two nodes. This can be implemented within a query using either the formula in Equation 2 (where A and B are arrays of all attributes of all outgoing edges of two nodes) or using the Cypher `algo.similarity.cosine` method [4].

**Equation 2: Cosine Similarity**

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

## 7. ANTICIPATED OUTCOMES

### 7.1 The Neo4j4Us System

The Neo4j4Us System shall accept a student’s planned curriculum and shall use this, in addition to Neo4j graphs and machine learning, to return the following outputs:

- Any constraint violations or errors present in the planned curriculum.
- An indication of the performance of other similar students with the same planned curriculum.
- An indication of the likelihood of the student qualifying for their degree in 3 years.

The constraint-checking and indication of past student performance will be implemented in a Neo4j constraint-checking tree and recommendation engine respectively. The likelihood of passing the degree will be determined by machine learning algorithms which use the same data stored in Neo4j. The design of the key features of the system (the constrain-checking tree, the recommendation engine and machine learning algorithm) are detailed in Section 4.

### 7.2 Expected Impact

The Neo4j4Us system will address the problems faced by students during registration. Using the system, the student is less likely to enrol in a planned curriculum which does not meet the requirements of their degree and can make better-informed decisions about their future plans (knowing the performance of similar students and their likelihood of passing). Students using this knowledge will make fewer errors when registering and will potentially avoid a degree or course which may not be suited to them (thereby avoiding failure and extending their degree). The inferred likelihood of their qualifying in 3 years will also be shown so that those who need to plan their degree over a longer period will do so.

### 7.3 Measurement of Success

The project will be successful if it can detect violations of all the main rules of a BSc curriculum, identify students whose plan makes them very similar to those who performed poorly (e.g. failed to meet readmission requirements), and can predict students unlikely to complete the BSc(CS) in 3 years to an acceptable degree of accuracy.

The success of the Neo4j constraint-checking tree and recommendation engine will be measured by extensive testing and accuracy comparisons (as outlined in Sections 4.1) respectively. The measurement of accuracy, precision and recall within the Bayesian Network predictions will indicate the success of the machine learning aspect of the Neo4j4Us system. The system will be successfully completed if a student may gain useful, accurate insights based on their planned curriculum.

## 8. PROJECT PLAN

### 8.1 Risks

A detailed risk matrix specifying the possible barriers to the success of this project, ordered by risk factor, can be found in Table 3 in Appendix B.

### 8.2 Timeline

The timeline corresponding to the project deliverables outlined in Section 8.5 can be seen in the Gantt chart in Figure 1 in Appendix A.

### 8.3 Resources

No specialised equipment is required to complete this project. All graph databases, queries and wrapper classes will be implemented in Neo4j and Java on a capable laptop. Machine learning will require development in Python or Java on a similar standard laptop. Student transcript data is required in the development of this project as all databases built and predictions made require this data.

### 8.4 Milestones

The following milestones represent the significant tasks to be accomplished throughout the duration of the project. These milestones can be viewed in connection with project deliverables and timeline in the Gantt chart (Figure 1, Appendix A).

#### *8.4.1 The Advisor Component – Josh*

1. Clean and prepare anonymised transcript dataset for insertion into Neo4j. (2 days)
2. Import transcript data into Neo4j and correct any erroneous node-edge relationships. (2 days)
3. Create constraint-checking tree and Java program to check that students do not violate any course constraints. (4 days)
4. Create queries to find similar students and determine their past performance. (6 days)
5. Create a wrapper program to facilitate the input and output of the decision tree and graph queries. (2 days)
6. Import the Gradebook data into a SQL database and translate the Neo4j queries to SQL. (2 days)
7. Compare the performance of Neo4j and a SQL database (6 days).
8. Extend the constraint-checking tree to identify smaller violations. (2 days)
9. Experiment with various approaches to grouping similar students, as mentioned in 4.1.4. (3 days)

#### *8.4.2 The Predictor Component – Edwin*

1. Create the Machine Learning system base. (24 days)
2. Integrate machine learning system with the Neo4j database in preparation of training the system. (4 days)
3. Train the system using a Neo4j database. (7 days)
4. Evaluate the accuracy of the predictions of the machine learning system. (5 days)

#### *8.4.3 Joint Responsibilities*

1. Merge the functionalities of the constraint-checking tree, Neo4j query results and machine learning outputs into a unified system. (1 day)
2. Perform validation testing with on the combined, integrated system. (3 days)

### 8.5 Deliverables

Deliverables after the submission of the Final Project paper are detailed in the Gantt chart in Figure 2, Appendix A.

**Table 2: Project Deliverables & Timeline**

| **Deliverables** | **Start date** | **End date** |
|---|---|---|
| Literature Review | 15/3/2019 | 2/4/2019 |
| Literature Review draft | 15/3/2019 | 25/4/2019 |
| Literature Review final | 25/4/2019 | 2/5/2019 |
| Project Proposal | 28/4/2019 | 23/5/2019 |
| Project Proposal draft | 28/4/2019 | 15/5/2019 |
| Project Proposal revision | 15/5/2019 | 21/5/2019 |
| Project Proposal Final | 22/5/2019 | 23/5/2019 |
| Project Proposal presentation | 15/5/2019 | 29/5/2019 |
| Development | 1/7/2019 | 11/8/2019 |
| Clean anonymised dataset | 1/7/2019 | 5/7/2019 |
| Neo4j Graph creation | 5/7/2019 | 7/7/2019 |
| Create tree for recommendation functionality | 7/7/2019 | 13/7/2019 |
| Machine Learning implementation | 1/7/2019 | 25/7/2019 |
| Trained Machine Learning system | 25/7/2019 | 2/8/2019 |
| Integration of systems | 2/8/2019 | 7/8/2019 |
| Validation test results | 7/8/2019 | 11/8/2019 |
| Demonstration | 15/7/2019 | 16/9/2019 |
| Feasibility demonstration | 15/7/2019 | 19/7/2019 |
| Code submission | 31/7/2019 | 1/8/2019 |
| Final Demonstration | 16/9/2019 | 16/9/2019 |
| Report | 1/8/2019 | 26/8/2019 |
| First draft | 1/8/2019 | 10/8/2019 |
| Second draft | 10/8/2019 | 20/8/2019 |

|  |  |  |
|---|---|---|
| Final project paper revision and hand in | 20/8/2019 | 26/8/2019 |
| Reflection Paper | 26/8/2019 | 7/10/2019 |

## 8.6 Work Allocation

The aspects of this project completed individually and in collaboration are outlined in Sections 8.4.1, 8.4.2 and 8.4.3 which reflect the milestones of the project.

## REFERENCES

[1] Chabane, S. et al. 2018. *Occupations in high demand in South Africa*.

[2] Dynamic Rule-Based Decision Trees in Neo4j | Max De Marzi: 2018. *https://maxdemarzi.com/2018/01/14/dynamic-rule-based-decision-trees-in-neo4j/*. Accessed: 2019-05-01.

[3] Gorakala, S.K. 2016. *Building Recommendation Engines*. Packt Publishing Ltd.

[4] Needham, M. and Hodler, A.E. 2019. *Graph Algorithms*. O'Reilly Media, Inc.

[5] Running Decision Trees in Neo4j @ Neo4j GraphConnect 2018: 2018. *https://neo4j.com/graphconnect-2018/session/decision-trees-in-neo4j*. Accessed: 2019-05-01.

[6] Tsamardinos, I. et al. 2006. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*. 65, 1 (Oct. 2006), 31–78. DOI:https://doi.org/10.1007/s10994-006-6889-7.

## APPENDIX A

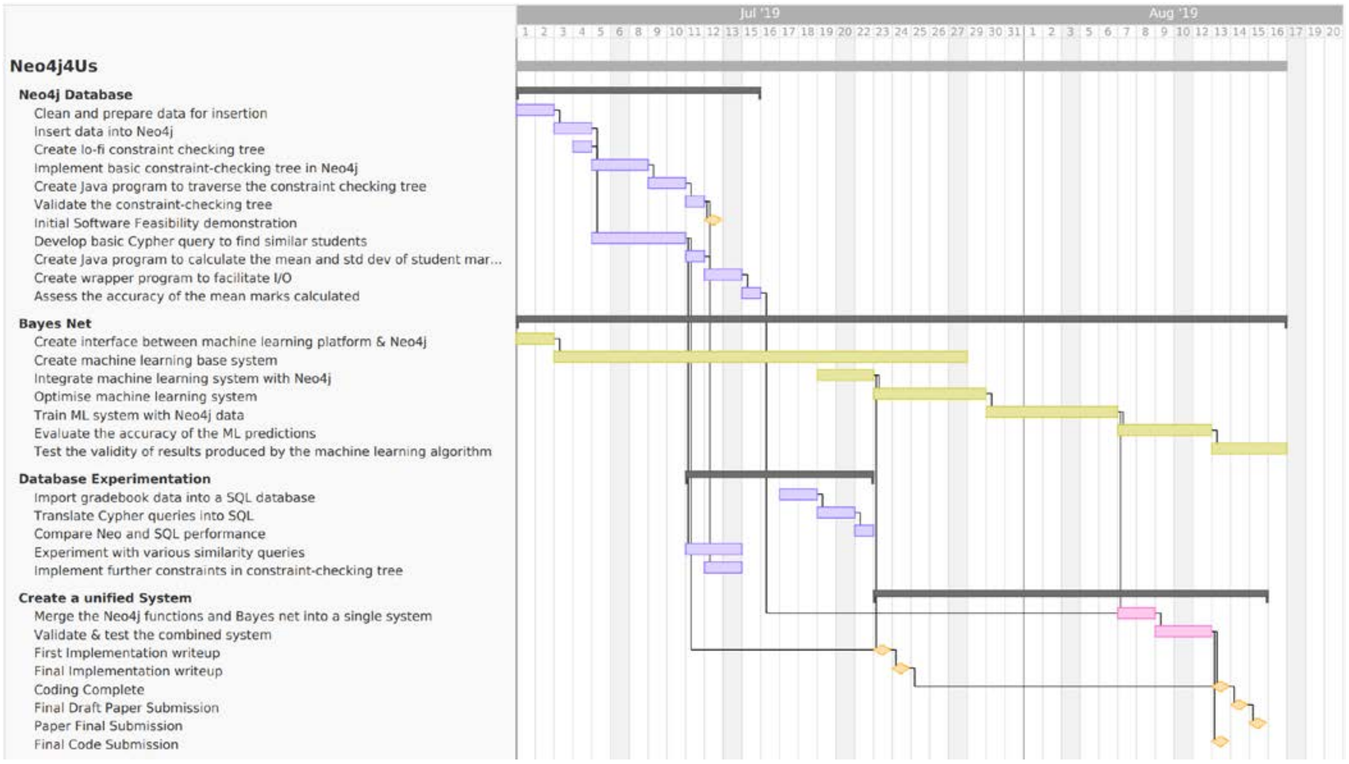


**Figure 1: Gantt Chart for Project Milestones**

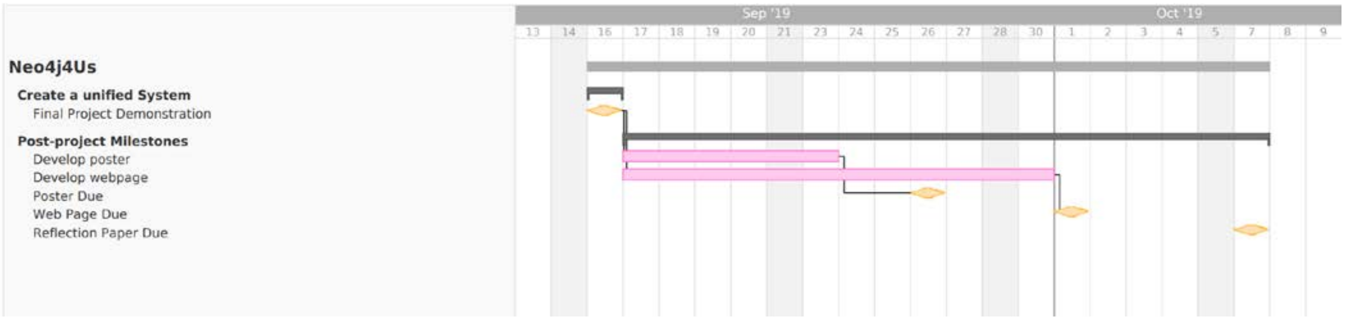


**Figure 2: Gantt Chart for Post-Project Deliverables**

## APPENDIX B

**Table 3: Risk Matrix**

| **Risk Description** | **Probability (1-10)** | **Impact (1-10)** | **Risk Factor** | **Consequence** | **Mitigation Strategy** | **Monitoring** | **Contingency** |
|---|---|---|---|---|---|---|---|
| Creating and comparing a Neo4j and SQL implementation of the same system may take too much time, and neither will be fully completed. | 6 | 8 | 48 | Conclusions regarding the performance benefits of Neo4j cannot be made. | Complete the Neo4j implementation before attempting the SQL implementation. | Assess the time remaining available to implement an SQL database and determine the likelihood of successfully completing the implementation. | Investigate other alternative comparisons that could be completed in less time. |
| The incorrect inference method is selected producing inaccurate Bayes net predictions. | 6 | 7 | 42 | The required accuracy level of predictions cannot be met. | Test both the structure and results of the inference method in question regularly. | Monitor the progress of the implementation and note any obstacles to progress early on. | Implement another inference method from the list of proven methods in the literature. |
| Creating a basic Bayes net without the use of external libraries may not be feasible. | 6 | 7 | 42 | Much time will be lost in development, reducing the time for experimentation. | Build the system in accordance with best practises and standardised structures set out by the given textbook. | Monitor the progress of the implementation and note any obstacles to progress early on. | Use template code for the basic structure of the network. |
| Mastering Cypher and developing the correct queries for Neo4j dominates the project timeline. | 5 | 7 | 35 | Limited time would remain for the comparison with SQL and further experimentation with Neo4j queries. | Create and implement the queries on a smaller test database before attempting work with the full dataset. | Monitor the progress of a simplified implementation of the database. | Discuss other project directions with the supervisor. |
| Scope of the project evolves beyond what is detailed in the proposal. | 5 | 7 | 35 | The project does not complete, loses focus and no insightful findings can be published. | Refer continuously to the planned work in this proposal and confirm any changes in direction with the project supervisor. | Strictly adhere to the milestones and Gantt chart. | Determine which project aspects are inessential and terminate them. |
| Cleaning the data and importing into Neo4j delays the rest of the project. | 3 | 6 | 18 | The time available to complete other more critical aspects of the project become restricted. | Start work on importing the data as soon as possible and insert data in small batches. | Track the completeness of the database before the deadline set in the Gantt chart. | Postpone the insertion of the full data and work with a small set of manually entered Neo4j data. |
| Merging the Neo4j queries and Bayes Net becomes more complex than the development of each component individually. | 4 | 3 | 12 | A unified system is not produced. | Ensure regular communication within the team so each member understands the other's work. | Track the progress of implementation in team coding sessions. | Create individual classes to facilitate I/O (create 2 different systems). |
| Determining the best method to group similar students requires more complex testing than what is outlined in Section 4.1. | 4 | 3 | 12 | No conclusive results may be published regarding determining the similarity of students. | Confirm the methods of experimenting with academic staff. | Once the data from the experiments have been collected, confirm the methods of analysis with the supervisor. | Focus on the simple similarity query and SQL implementations instead. |
| Anonymised data is not approved for use in the Neo4j database. | 1 | 9 | 9 | A large and varied dataset will not be used for Bayes net training & Neo4j querying. | Submit the necessary ethics clearance if the data is not supplied. | Track communication with the supervisors until the data is obtained. | Create a script to generate a large and randomised simulated student transcript dataset. |
| The team collapses due to miscommunication or a member dropping out. | 1 | 9 | 9 | Many aspects of the project will go unfinished. | Each member should work only on their individual responsibilities to ensure that some work may be submitted. | Ensure regular communication through weekly meetings. | Re-evaluate the project direction with the supervisor and devise a new plan. |