Covenant Journal of Informatics & Communication Technology. Vol. 8 No. 2,December 2020
ISSN: *print* 2354 – 3566 *electronics* 2354 – 3507 DOI:

An Open Access Journal Available Online

Application of Genetic Algorithm to The Job Assignment Problem with Dynamics Constraints

Chika Yinka-Banjo[1], Ajao Kamal Abayomi[1], Precious Ifeanyi Ohalete[2]

[1] University of Lagos, Akoka, Nigeria
[2] Alex Ekwueme Federal University, Ndufu-Alike, Nigeria
cyinkabanjo@unilag.edu.ng, kamalajao@gmail.com, ohaleteprecious@gmail.com

Received: 10.08.2020 Accepted: 30.09.2020
Date of Publication: December 2020

Abstract—The process of giving out an assignment to an individual that results to delay, or non-performance of the job is from the cause of not evaluating the minimum cost of the work and the right person to perform the assignment. Assignment problem entails assigning a precise person or thing to an exact task or job. The optimal result is to assign one person to one job. The most common method to solve assignment problem is the Hungarian method. In this paper, Genetic Algorithm is applied to solve assignment problems to attain an optimal solution. The "N men – N jobs" issue is the core task issue, where the general expense of tasks is limited as a result of allocating a single job to just an individual. In deciphering this issue, Genetic Algorithm (GA) and Partially Matched Crossover (PMX) are been utilized as an exceptional encoding plan. GA was evaluated alongside the Hungarian method and the results clearly showed that it performed better than the Hungarian method.

Keywords/Index Terms—Assignment problem, Branch and Bound Technique, Constraints, Genetic Algorithm, Hungarian assignment method.

1. Introduction

Organizations in modern-day businesses have a greater need to handle their activities efficiently to pull off competitive advantages in every part of the organization. To attain this, jobs need to be assigned to the best-qualified personnel.

Successful organization is an engine; the work parts are simply unsuccessful if they do not fit exactly where they are put. Likewise, putting the wrong people in mission-critical positions can be expensive and counterproductive to business performance and organizational development. Maintaining and improving the efficiency and effectiveness of your employees are important to any business' growth and success, hence the justification for solving the problem of assignments (Elsayed et al., 2010). The Optimization of jobs provides call recording tools, agent assessment, and coaching, performance management, planning and scheduling, contact tracking, and feedback surveys. These technologies are implemented in applications that allow companies to leverage customer interaction data to increase efficiency, improve sales efforts, and ultimately improve customer experience. Workforce optimization helps companies to improve the efficiency and efficacy of their customer interactions by collecting interactions across all platforms, extracting lessons, and taking decisions that affect business outcomes and customer experience. (Naveh, et al. 2007). In every organization, Human resources seem to be the bedrock, as its involvement in operations, such as agents in client care, clinical delegates in visiting doctors, producing laborers, contact focus faculty in client cooperation, aircraft group in client assistance, explicit heads, and so forth.

Efficiently controlled employees eventually contribute to the organization's efficiency, and thus effectiveness. Workforce application intends to cover the workload with the available resources while respecting work constraints, balancing the workload among employees, and minimizing dormant time. Significant benefits are being provided in enhanced workplace management and preparation, such as increased efficiencies, reduced costs, enhanced customer service, and greater employee satisfaction, (Naveh, et al. 2007).

In linear programming, issues of assignment are distinct as delegated persons are obliged to carry out assignments. The assignees, for example, could be workers who should dole out work tasks. In assigning jobs, the major and standard issue is the allocation of jobs to individuals. To fit the meaning of a task issue, these sorts of utilizations should be defined in a way that meets the accompanying assumptions.

a. Assigned individuals and quantity of tasks are respectively indistinguishable. (n indicates such numbers)

b. One mission assignment to exactly one assignee.

c. One assignee shall complete one assignment.

d. Costs are aligned with assignee i (i= 1, 2, 2,.,n) executing function j (j= 1, 2,., n).

e. The goal is to decide how all n tasks should be carried out to reduce the total cost.

In this work we applied GA to job assignment problem with constraints. In

what follows, Section 2 talks about some related literature. Section 3 explains the methods and algorithms adapted in this work. Section 4 shows the results tested in different case scenarios simulated and real environments. Section 5 concludes the study.

2. Literature Review

Garrett *et al* (2005) showcased the Genetic Algorithm (GA) towards overcoming the Sailor Assignment Problem (SAP) for the United States Navy. The SAP is a mind-boggling issue of assignment where every one of the n sailors must be allotted one task from an assortment of m employees.

The genetic algorithm adopted for the study is been connected to an existing algorithm, the Gale-Shapley algorithm, in building such assignments and giving observational outcomes indicated that the GA could create great arrangements with huge cost savings In Anwar (2017) the Hungarian algorithm was built specifically on using a graph in the general method. The approach was attained by selecting the minimum cost (edge) from the cost (edges) and eliminating the chosen edge as well as the edge-related nodes, then deleting all other node-related edges. The edges are the outflow of assigning people to jobs, the roles and entities are the nodes. A Constructive Genetic Algorithm (CGA) to the problem of genetic selection was presented in (Lorena *et al.*, 2002). Compared with a conventional genetic algorithm the CGA has several new technologies. These include diverse population sizes and an opportunity to use heuristics. With seniority and task priority restrictions (Caron *et al*., 1999) proposed a solution to the work allocation issue where seniority requirements allowed the approach to be such that no unassigned person can be given a job except the same or higher seniority appointed individual is unassigned. Priority requirements state that the compromise must be such that no unassigned job can be allocated without a position being unassigned with the same or higher priority. Bogomolnaia and Moulin (2001) found a situation where all actors have specific expectations and suggested the method of Probabilistic Serial (PS). They describe a new notion of output, named ordinal performance, and show that the probabilistic serial process considers a normally effective assignment free from envy. However, their algorithm is crucial to the limiting expectation of specific expectations. The method used by the author was based on a reinterpretation of the PS system as an iterative algorithm for computing a flow in a related network. It was shown that seeking a random assignment that is both normally effective and envy-free is unlikely for even a poor strategy evidence mechanism on the maximum choice domain.

Semih *et al*., (2008) applied the distribution-type warehouse assignment question where different types of goods were acquired from various vendors for storage in the warehouse for a specified period and allotment to different clients. Their analysis aimed to develop a layout of multi-level warehouse shelves that reduces the annual cost of transportation. Since the original mathematical model proved to be NP-hard, a Particle Swarm Optimization Algorithm (PSO) was developed as a novel heuristic to decide the optimal configuration.

GA has been applied in measuring future

progress in economy and market by refining the rules-lists constructed for fuzzy logic control after the removal of inherent redundancy (Alfa et al., 2019 & 2020).

A variant of GA with selection and evaluation was implemented for a timetable scheduling problem by replacing crossover and mutation using tabu search memory and course sandwiching (Abayomi-Alli et al., 2020).

2.1 Genetic Algorithm (GA)

is a heuristic, natural evolution-inspired search, and optimization strategy (McCall, J. 2005). It was first conceived by John Holland and later developed by numerous researchers. Optimization or solution-searching is a heuristic technique of GA, at first propelled by the Darwinian hypothesis of development by (genetic) preference. Theoretical interpretation of the evolutionary cycle is used by GA in creating an answer to issues. Every GA works on artificial chromosomal groups.

On each population (generation) of chromosomes (individual solution) generated, three operations are performed:

a) Reproduction: In this process, individual strings are been matched their fitness functions respectively (Total cost function is assumed here).
b) Crossover: This is the mechanism of switching the two-string material at some point(s) with a chance through several chromosomes.
c) Mutation: This is the flipping component of the enthusiasm inside a chromosome at a specific area in an arrangement with a low (Sahu & Tapadar, 2007).

With a start-up of a chromosome sample which is randomly generated, a wellness-based selection and recombination cycle performed by the GA to make the following cluster, a replacement/successor group. In the cause of recombination, parent chromosomes are chosen, and the obtained genetic material is recombined towards the build-up of the youngster's chromosome. These will at that point move into the public of the replacements. A progression of progressive ages develops as this cycle is iterated, and the absolute strength of the chromosomes will in general increase until some stop prerequisite is met.

This selection process guarantees compatibility of the system with the darwinian survival of the fittest in the natural world, by transferring a
higher portion of the best-suited genes
to the next generation
(Sahu & Tapadar, 2007).

2.2 Genetic Algorithm Crossover Variants

The Job assignment problem is likewise a Constraint Optimization Problem (COP) given the task of machines or occupations dependent on status, skill level, availability, etc. as mentioned in the introduction of this study. There are several variants of the GA that are applied to different COP. Blend Fusion (BLX-α), Simulated binary crossover (SBX), Simplex Crossover (SPX), Parent Centric Crossover (PCX), Triangular Crossover (TC), and Partially Matched Crossover (PMX) are instances of GA variations. These variations chiefly contrast in their utilization of crossover operators and mutation operators. (Elsayed et al., 2010)

3.0: Methodology

The purpose of this study is to apply Genetic Algorithm Optimization to work assignment question N workers on M computers, where N is the number of jobs allocated, and M is the number of staff. This study will use Ikeja Electric Distribution company based in Lagos as case study, create a mathematical model and adopt an objective optimization algorithm to obtain optimal job assignments. The methodology adopted is the bottom-up approach to increment development. This kind of application commences with an architectural program design. Bottom-up implementation begins with the system's lowest-level components. The method of a job assignment is represented as a series of strings corresponding to each

the base 10 value of the bit representation of that row e.g.

<4 2 1> = <1 0 0, 0 1 0, 0 0 1>; where each row is separated by a comma. Each permutation of a given string is a valid solution therefore, contains the best solution. Given this encoding scheme generates N! strings where N is the number of alleles in a chromosome, it is correct to assume that each solution corresponds to a character encoding. The implementation from bottom-up begins with components that use everything else but use nothing on its own. Firstly, a testbed for each component is built and secondly components are grouped into subsystems after having been evaluated equally when the lowest level components have been tested using a testbed. The process continues until the entire system is fully completed and then checked as a whole.

3.1 Job Assignment Problem

Given N computers and N people. Mathematical representation, with the following symbols, can be used to describe it:

i →row number denoting ith man i ε [1, N]

j →column number denoting jth machine j ε [1, N]

Where [1, N] is an infinite set of natural numbers

C[i][j]→ cost of assigning jth machine to ith man

X[i][j] = 1 if jth machine is assigned to ith man = 0 otherwise

i.e. on the job assignment matrix if jth machine is assigned to ith man, the element that corresponds to the ith row and jth column = 1, else it will = 0.

The question can be posed as follows:

Minimize the total cost function

$$\sum_{i=1}^{N}\sum_{j=1}^{N} C[i][j]X[i][j]$$

Subject to the following constraints:

$$\sum_{i=1}^{N} X[i][j] = 1 \;\forall\, j = 1, 2 \ldots N \qquad (1)$$

$$\sum_{j=1}^{N} X[i][j] = 1 \;\forall\, j = 1, 2 \ldots N \qquad (2)$$

$$X[i][j] = 1 \text{ or } 0 \qquad (3)$$

This indicates that the ith man for all instances of (1..N) (natural numbers), there must be a jth machine for all (1…N) on the job assignment matrix.

3.2 Job Genetic Algorithm approach

1. After encoding the solution strings, the "Binary tournament selection" method is

adopted for population crossover selection.

2. In the binary tournament selection, two strings are selected and compared randomly, the optimal one chosen for parenthood is replicated M times. Where M is the population size.
3. Then the Partially Matched Crossover (PMX) is employed for crossover, which can be better explained by the following example. Two strings are chosen for crossover <1 3 4 2 5> and <2 1 3 5 4>. Arbitrarily, two numbers (positions) among 1 and L are created where L is the length of the string, here L=5. The swath of genetic material (alleles between two points on the chromosomes) from one string, and the corresponding swath from another string, between the selected boundaries of the chromosome, are interchanged such that

3 ↔ 1 4 ↔3 2 ↔ 5, which implies1↔4 and 2↔5 as shown in Figure 1.

1	**3**	**4**	**2**	**5**
	↕		↕	
2	**1**	**3**	**5**	**4**

FIGURE 1: PMX CROSSOVER

On the off chance that 1 in the part outside the two-hybrid focuses is subbed by 4 and 2 in the segment above, 5 will remove the two crossover points. Switching the alleles outside the selected boundaries of the two strings ensures the strings produced by the PMX crossover are valid.

In Figure 2, the PMX crossover approach adopted ensures that crossover generated strings are valid possible combinations of < 1 2 3 4 5 >.

4. After mixing, a population exists that contains the parent population and children population. The fittest individuals are selected from this generation for the next iteration.

4	**1**	**3**	**5**	**2**
5	**3**	**4**	**2**	**1**

FIGURE 2: PMX CROSSOVER OUTCOME

Two methods could be used for the selection process:

a. A method is devised, which orders each individual in this population in ascending performance order (objective function value), the string encoding the lowest total assignment expense will have the maximum objective function

benefit. A fixed number of individual strings can be picked from this population under each group, in which each population can be split into 4. For instance, it is assumed that the string values are normally distributed with a mean value of μ and a standard deviation of μ, and then the population could be classified into four categories: those with average values above -μ + 3*π, those with average values between μ + 3*μ and μ, those with values between μ and μ-3*π, then those with values above average. Those below average have values

below μ — 3*ÿ. In this way the ethnic composition is preserved.

b. One way of choosing the

population is to store the string with the best objective function value individually in an array at each iteration, and then equate iteration with the best string of the population. The strongest string cannot avoid this path. The GA has a drawback of converging to a local equilibrium i.e., a premature convergence that contributes to a successful but not the best solution. The solution adopted to minimize this weakness is to maximize the population size at each iteration and to maintain a demographic diversity at each iteration.

5. A mutation alternative, Inversion, is then implemented which selects two random spots in a string and flips the corresponding values at that location.

3.3 Job Functional Requirement

The real essence of a design definition is the practical specifications of the functionalities of the framework that satisfies the requirements specification of this project. This is done by going through the applications functionality as specified in its specification. For each requirement, a more generic function had to be specified that could implement these functionalities. These requirements can also specify what the developing system should not do.

1. Valid Dataset

   This is a user requirement which requires that the dataset uploaded to the optimizer is compatible with the problem set and conforms with the scope of this study, which is the activities of staffs in the ICT unit at Ikeja Electric Distribution Company whose branches are located in Lagos. This implies that the dataset would contain schedules of possible activities executable by these ICT staff, organized by time stamps and expected duration of executions.

2. Simulation Manager

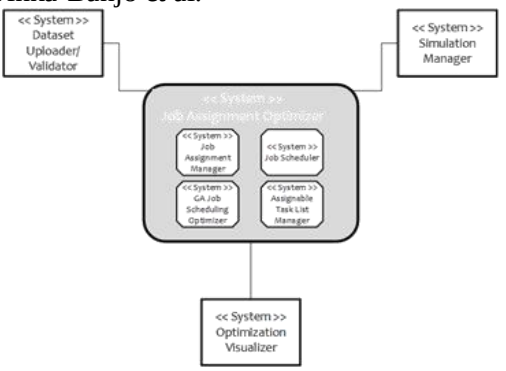
   This system requirement ensures that the system simulates the efficiency of a given dataset, either optimized or unoptimized. Each simulation is timed, allowing the tester to benchmark the dataset through visual inspection of the animation during simulation.

3. Optimization Visualizer

   This system requirement visualizes on a line graph the comparison between multiple solution iteration using Key Point Indicators (KPIs) such as the fittest individual in every generation and population size at each iteration (i.e. alternated to escape local optima).

4. Job Assignment Optimization

   This system requirement employs the adopted GA algorithm to optimize the inputted dataset. This optimizer minimizes the objective function of the job assignment problem as defined in this study constrained by the intrinsic priority of a task, skill-level, staff's performance on that job, location from task, and staff's current workload. Figure 3 show as explained.



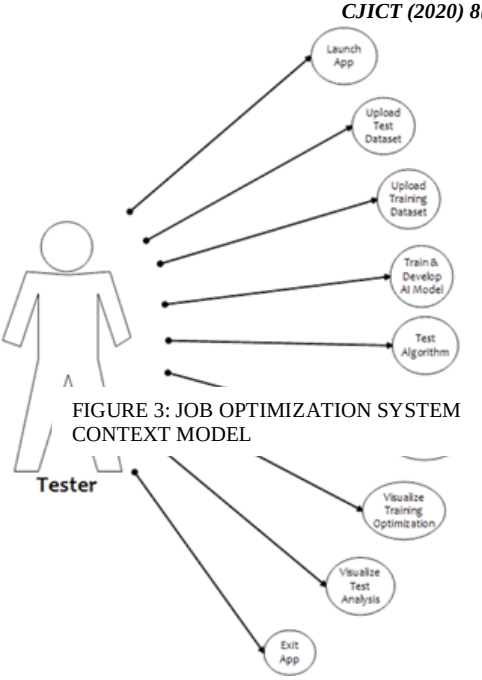


FIGURE 3: JOB OPTIMIZATION SYSTEM CONTEXT MODEL

FIGURE 4: TESTER USE CASE

Figure 4 shows the activities on how a tester of this system interacts with the developed system.

3.4 System Design

This section introduces system models that show with a varying degree of abstraction, different perspective of this system used to design the developing system.

3.4.1 Context Model

The design model in Figure 5 is used to illustrate device integration with an external perspective of the framework and its environment. This model helps to reflect the system's borders and its setting. Such boundaries could be established to accommodate technological as well as non-technical considerations such as social and organizational issues. After the boundaries have been ascertained, the research done will attempt to determine the nature of that modelled sense and the dependencies the device has on its setting.

1. Job Assignment Manager manages the job assignment list while considering other constraints such as seniority constraint, staff availability constraint, job role constraint, skill-level constraint etc.
2. Job Scheduler converts the information uploaded via dataset to various job assignments on the job list of different ICT staff on the system.
3. GA Job Scheduling Optimizer employs the GA strategy adopted in this study to minimize the objective function of the job assignment problem as defined in this study. It operates on the

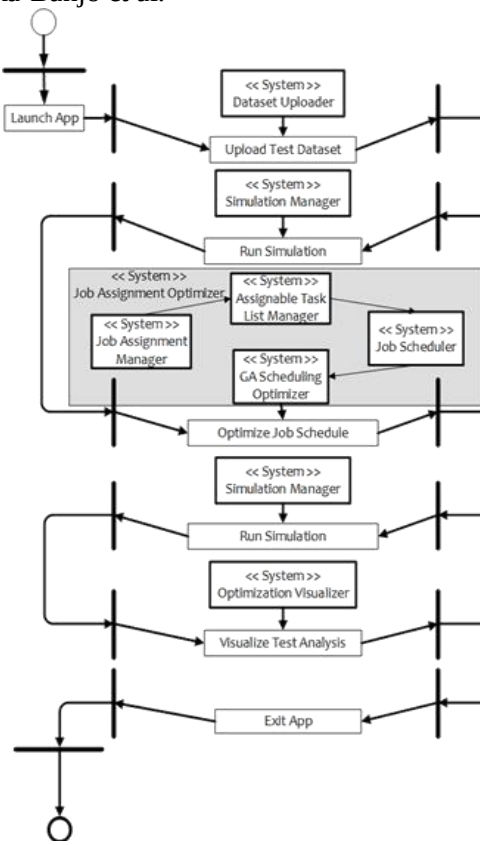


FIGURE 5: JOB OPTIMIZATION SYSTEM ACTIVITY MODEL CONTEXT MODEL

` uploaded dataset to generate an optimal job assignment that will be executed by every staff of the ICT unit in the shortest time.

4. Assignable Task List Manager enforces the job role constraint, ensuring that staff in the simulation system only undertake jobs that are assigned to them. This subsystem stores the list of assignable jobs as it concerns each job role captured on this system.
5. Dataset Uploader/Validator enables the system tester upload dataset for optimization. This system component analyses the

uploaded dataset to ascertain its validity and its conformity to the various job assignment constraints as defined in this system such as operational hours,
job assignments based on job roles, etc. The uploader formats the contents of a dataset to extract assignable jobs, jobs to be done, owner of a job, subject of a job, object of a job, which is stored on database before processing.

6. Simulation Manager simulates the performance of a job assignment, optimized or unoptimized. It enables the tester to visualize in real-time through animations, the pseudo execution of operations as defined in each job assignment.
7. Optimization Visualizer shows the results of the optimization process as adopted in this project. It uses a line g
tl[illegible]
test scenarios.

3.4 Process Model

In accordance with the background model, the System model in Figure 5 is used to describe human and automatic systems in which each software program is used. The aim of this diagram is to display the actions that form a machine process and transfer of control from one operation to another. A filled circle indicates the start of a process, the end of a filled circle inside another Circle. Round-cornered rectangles represent activities, that is, the specific sub-processes that need to be performed. The operation diagram can include artifacts to show various systems that are used to support different

processes.

4.0 Test Case Scenario

This project focuses on the power distribution sector and the activities of the IT department as a service department to the other personnels and departments in the company scattered across seven business units in Lagos State. These units are used as test cases for simulation and test for efficiency of the genetic algorithm in relation to the job assignment problem. The algorithm would work on pseudo-data, built based on the job functions performed by each specialist, and benchmarked using the following parameters: Flexibility, Robustness, Optimality, Speed.

To enable proper analysis of data captured despite variability, the optimization process is simulated several times and recorded. Each run will contain 5 iterations corresponding to 200, 400, 600, 800 and 1000 job assignments. Each assignment is randomized to create a more realistic scenario.

The Hungarian algorithm and its variants are known as the most adopted approach for tackling the job assignment problem. The Hungarian algorithm is used to benchmark the adopted algorithm.

4.1 Data Dictionary

This section describes the test dataset for the job optimization problem as it concerns operations of staff in the IT department, as the technical support for other departments, maintaining sectors operations as it concerns IT infrastructure such as networking infrastructure, PCs, revolving doors, electronics and building electrical infrastructure. Table 1 shows a few samples of the job objects, job roles and job names as stored on the database defined in the analysis and design section of this project. Job objects - items around which a job is to be performed/object of the job execution e.g., network router, mouse, keyboard etc. These objects could be attached to a defining noun such as person or place e.g., Henry's mouse, Ayo desktop monitor etc.

Job roles – a grouping of staff based on department and their specialized.

Job names – name of each performable job of an IT staff of Access bank as captured by the system.

Table 1 represents the core data, not the entire data captured on the test dataset, which also includes username, branch name, branch location, skill-level etc.

TABLE 1 DATA DICTIONARY

Job Role	Job Object	Job Name
Network Technician	Main Hall Router, Customer Service Hall Router, Network Cable	Fix the router in the customer hall. We can't get internet at our office (Audit). My computer cannot access the network.
IT Technician	Revolving door, AC, Socket, faulty wire	The left revolving door at branch A is temporarily out of service. The socket at my station is faulty.
IT Support	Internet connection, Computer	I can't login to my window's account. My computer is not coming on.
Software Support	CIS App, Mobile App	I can't login Meter reading software. I can't balance my sheets for today.

Job names as mentioned above accounts for the fact that job assigners on this system are not IT savvy (not familiar with IT terms), as is the case in many organizations, therefore, adopting a natural vocabulary instead of technical vocabulary. Job names are dynamically bound to the job assigner's location and username, therefore, the system

can permit the use of pronouns such as I, my, our, us etc., to make the job assignment name which makes it more natural.

4.2 Sample Input/Output/Results

Development of a JavaFX Application using Scene Builder for GUI, allowed the application user interface to be developed first. The components of the user interface determine how each aspect of the project object and scope is achieved. After the user interface was developed using scene builder, then different logic for manipulation of information within each component of the application where implemented in their controller classes.

JavaFX is the new object-oriented framework for developing Java GUI programs. Every JavaFx application must extend the abstract javafx.application. .Application class, which defines the essential framework for writing JavaFx programs. Each JavaFx application must have a static launch method, a static main method, start method, a scene and a stage. The launch method defined in the Application class is used to launch stand-alone JavaFX applications. The main method is a method defined in the Object class which is inherited by every Java application. The main is method called when a java application is executed. The start method also defined in the Application class, and must be overridden by its concrete subclass.

A solution was to adopt a framework that is capable of seamlessly loading and displaying a desired layout container from a set of layout containers loaded. It copies the root layout container of the

requested Scene to the displaying window. This framework uses a hash-function to map an identifier as key and the layout container as the object. This framework is an imitation of the framework developed by a Java Evangelist at Oracle, Angela Caicedo.

Every GUI feature has its own window (a file with an extension of.fxml) and controller (a file with an extension of.java). FXML is an XML extension used to create a markup language to manage the layout of a web development JavaFx Framework, associated with HTML. Controller groups manage user activities including events caused by the mouse, key events triggered etc.. In the corresponding parts the GUI components and their controller groups would be described.

4.2.1 ***Inputs***

This portion explains the application software built for this project and how it meets the purpose of the project, in effect the core components and inputs gathering knowledge through the operator's contact with the simulation device. Below is a description of GUI Components and their set of controllers: DBConnect.java, GA.java, HA.java, Jobs.java Vertex.Java, WeightedEdge.java, Graph.java, Project.Java, DataUpload.fxml & DataUploadController.java,

SimulationPage.fxml & SimulationPage.java

1. DBConnect.java class is the communication module between the system and the database.
2. GA.java class implements the GA modeled for the optimization of the job assignment problem. This class contains a mutation operator (inversion), fitness evaluator for local and global best, and configurable crossover possibility to adjust the efficiency of the algorithm.
3. HA.java implements the popularly adopted Hungarian Algorithm used to benchmark the performance of the GA adopted for this project. During testing, the same values passed to the Genetic Algorithm is passed to the Hungarian Algorithm to ensure the benchmark results are accurate.
4. Jobs.java class models the single core entity of this project. It stores other variables that describe it properly such as job assigner username, job name, job object, time of assignment etc. This object is instantiated when the test data uploaded to database is retrieved. This object is randomized for each test run to simulate the randomness of job assignment in real-life scenarios.
5. Vertex.java class models the java object that represents a node on a graph. These nodes represent the bank branches. The coordinate location of each node is preconfigured before program execution to model the location.
6. WeightedEdge.java class models the rail lines between way stations. The weight of each edge represents the distance between any two branches and distance between stations. it is represented on the program as a weighted edge with a cost, calculated by the system on execution as a straight line between two points.
7. Graph.java class models the branches of the bank as a connected unit that interoperate, therefore, can have IT staff shuttle several branches on job assignments. The class models the branches as undirected connected graphs of the transportation network between branches of the bank. After the system has optimized the uploaded job assignments, the new scheduling and routing algorithm is simulated on the graph object and displayed on the simulation page.
8. Project.java class is the main class for this project's application. It contains the main method, start method, Stage object, scene object and the launch method required to execute the JavaFX application. In this class all layout containers are loaded and the display container set.
9. DataUpload.fxml& DataUploadController.java

DataUpload.fxml is the GUI module (fxml file) that contains the GUI components, where data processing is controlled or visualized. DataUploadController.java is the controller class for DataUpload.fxml. This class uploads test data to database, which is then retrieved for job optimization. This class controls the main page for simulation and simulation configuration. It contains the layout containers for the simulation planner panel and graph panel, which are loaded using

Angela Caicedo's Framework, to display the requested panel stored in the display container set. Figure 6 is the upload testdata interface develop for the application.

FIGURE 6: UPLOAD DATA PAGE CONTEXT MODEL

10. SimulationPage.fxml & SimulationPage.java
SimulationPage.fxml are the GUI modules (fxml file) that contain the GUI components for visualization and the simulation of the unoptimized and optimized job assignments in the application.
SimulationPageController.java is the controller class for SimulationPage.fxml.

4.3. Results and Discussion

The genetic algorithm along with other optimization algorithms varies in performance at each test instance due to unpredictable and uncontrollable factors such as distance between the initial state and optimal state etc. Other factors that can alter the runtime are the complexity of the dataset as regards to defined constraint of the job assignment problem. For this reason, one case scenario is inputted to the system and optimized twice using a genetic algorithm before data is recorded, to test the robustness and flexibility of the adopted approach. Figure 7 shows the result of the first test after running multiple assignments. The tables on the right show how many times an optimal

solution was found by each algorithm at every optimization iteration run.

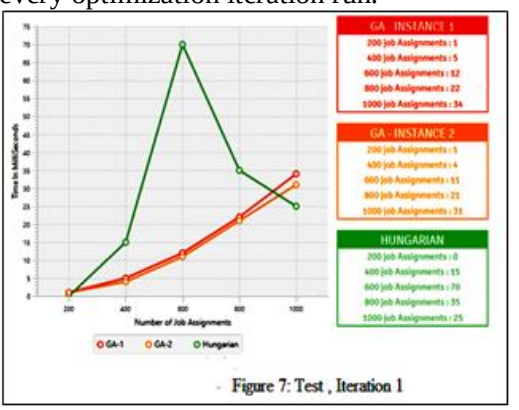


Figure 7: Test , Iteration 1

Figure 8 shows the result of the second test after running multiple assignments. The tables on the right show how many times an optimal solution was found by each algorithm at every optimization iteration run.

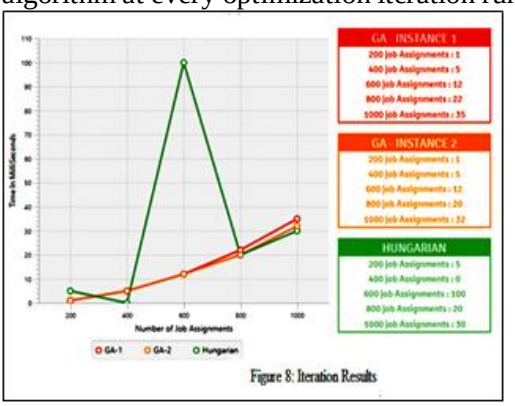


Figure 8: Iteration Results

Figure 9 shows the result of the third test after running multiple assignments.

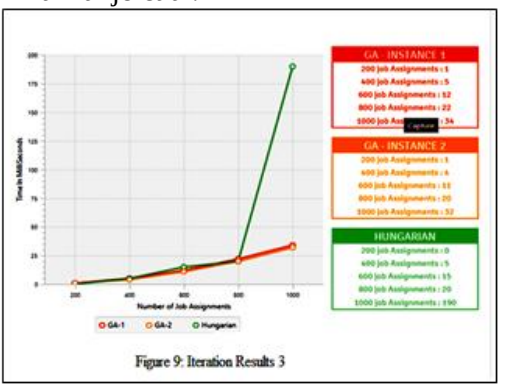


Figure 9: Iteration Results 3

Figure 10 shows the result of the fourth test after running multiple assignments.

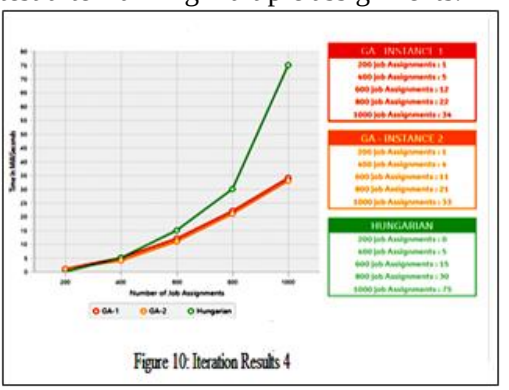


Figure 10: Iteration Results 4

Figure 11 shows the result of the fifth test after running multiple assignments.

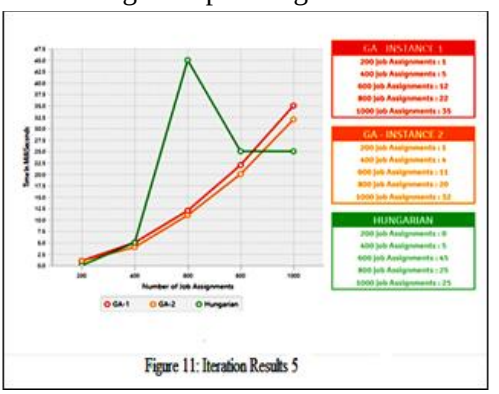


Figure 11: Iteration Results 5

Figure 12 shows the result of the sixth test after running multiple assignments.

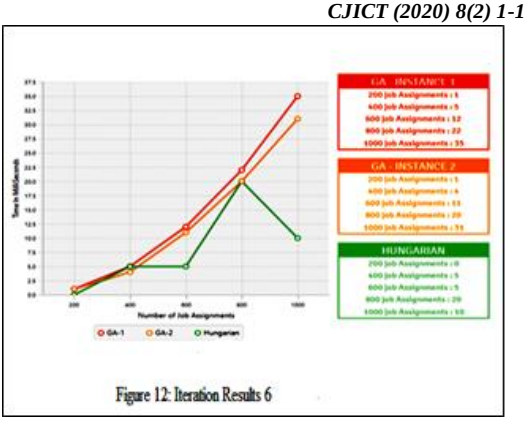


Figure 12: Iteration Results 6

As observed from Figures 7 - 12 above, the graphs illustrate that the performance of genetic algorithm is better than the traditionally adopted Hungarian approach. After every iteration, the fitness value of the optimal solution at each optimization run of a given job assignment is recorded. After each job optimization run, the fittest value is ranked, and the fittest is believed to be the optimal solution for that job optimization iteration i.e., for 200 job assignments, 400 job assignments, etc. The tables on the right show how many times an optimal solution was found by each algorithm at every optimization iteration run. E.g., Figure 7 -test iteration 1, the Hungarian algorithm did not find the optimal solution for 200 job assignments iteration, etc. Table 2 illustrates the performance of Genetic algorithm and the Hungarian algorithm

To achieve this level of performance by genetic algorithm, the crossover parameters and mutation parameters were optimally configured after several test iterations. The values were slightly changed and observed for performance gain till no further performance gain could be reached.

All in all, an exploratory examination into taking care of the Assignment problem utilizing Genetic Algorithm is

introduced. A scope of boundaries upsetting the algorithms is examined and their result in combination with the final optimum solution is been displayed.

The Genetic algorithm is easier and faster than the famous Hungarian method and the concluding output of the proposed method is way better than the Hungarian method just as observed from the iterated results obtained. After the completion of each job optimization run, the fittest value is ranked, and the fittest among the fittest is believed to be the optimal solution for that job optimization iteration.

TABLE 2: PERFORMANCE EVALUATION OF GA

	Genetic algorithm vs. Hungarian Algorithm
Flexibility	The Hungarian approach lacks consistency wherein it finds an optimal solution in a relatively shorter time than the genetic algorithm, and other times it's considerably slower than the genetic algorithm, making it inconsistent with random job assignments and not realistic for real-life application.
Robustness	As observed in the test result images above, the genetic algorithm scaled uniformly as job assignment counts increased. This uniformity shows that Genetic Algorithm is more robust for real-life applications.
Optimality	In the right column of the test result images, it can be observed that both instances of the genetic algorithm optimizations established the optimal solution more frequently than the Hungarian algorithm.
Speed	The Genetic Algorithm finds the optimal solution more frequently than the Hungarian Algorithm, observing the average result of the six test images above, it can be concluded that the genetic algorithm is relatively faster than the Hungarian approach.

5.0 Conclusion

Developing genetic algorithms and defining parameters for optimal performance is a big challenge. The PMX (Partial Mapping Crossover) genetic algorithm is superior to the standard Hungarian algorithm for work planning problems.

This performance is attributed to the crossing and mutation parameters used in the PMX algorithm. PMX divides each generation into four categories, grouped in ascending order. In each group, the best is used for cross mutations to ensure that the genetic algorithm avoids maximum local problems. This approach ensures that the genetic algorithm gets the best plan for the assignment problem. The algorithm is implemented in Java, which uses fork and join APIs for parallel processing, thereby increasing processing speed. It works because task scheduling can be modeled as a problem of division and conquest on genetic algorithms.

This article presents a genetic algorithm and applies it to work scheduling problems with dynamic constraints. As a result, two examples of the genetic algorithm were compared, and one of them was widely used in the new Hungarian method for work assignment problems. The results show that genetic algorithm is a better method for practical applications due to the dynamic constraints of speed, optimality, flexibility, and scalability. According to the results observed in Figures 7 -12, when there are fewer assignments (<200 assignments), the Hungarian algorithm and the PMX genetic algorithm are relatively good, but when there are more assignments (> 200) 600 works), the performance of the PMX genetic algorithm is better than that of the Hungarian algorithm. The results sometimes show spikes due to the complexity of software generated task assignments. This complexity is a

chromosome of initial solution generated in the form of a calendar, with a very low fitness value, which converges towards optimal results only near the stopping condition.

References

Abayomi-Alli, A., Misra, S., Fernández-Sanz, L., Abayomi-Alli, O., & Edun, A.R.,(2020). Genetic Algorithm and Tabu Search Memory with Course Sandwiching (Gats_cs) for University Examination Timetabling, *Intelligent Automation and Soft Computing*, 26(3), 385–396.

Alfa, A. A., Misra, S., Bumojo, A., Ahmed, K. B., Oluranti, J., & Ahuja, R. (2019). Comparative Analysis of Optimisations of Antecedents and Consequents of Fuzzy Inference System Rules Lists Using Genetic Algorithm Operations. In *International Conference on Advances in Computational Intelligence and Informatics*. 373-379.

Alfa, A. A., Yusuf, I. O., Misra, S., & Ahuja, R. (2020). Enhancing stock prices forecasting system outputs through genetic algorithms refinement of rules-lists. In *Proceedings of First International Conference on Computing, Communications, and Cyber-Security*. 669-680.

Anwar N. J. (2017). A New Method to Solve Assignment Models. *Applied Mathematical Sciences*, 11(54), 2663 – 2670

Bogomolnaia, A., & Moulin, H. (2001). A New Solution to the Random Assignment Problem. *Journal of Economic theory*, 100(2), 295-328.

Caron, G., Hansen, P., & Jaumard, B. (1999). The Assignment Problem with Seniority and Job Priority Constraints. *Operations Research*, 47(3), 449-453.

Chen, P. (1976). The Entity Relationship Model—Towards a Unified View of Data. *ACM Trans. On Database Systems*, 1 (1), 9–36.

Douglas, B. (2005). *Software Engineering for Students: A Programming Approach*, Fourth Edition, Pearson Education, United Kingdom.

Elsayed S. M., Sarker R.A., & Essam D.L. (2010). A Comparative Study of Different Variants of Genetic Algorithms for Constrained Optimization. *In: Deb K. et al. (eds) Simulated Evolution and Learning. SEAL 2010. Lecture Notes in Computer Science*, 6457, 177-186

Garrett D, Vannucci J, Silva R, Dasgupta D & Simien, J. (2005). Genetic Algorithms for the Sailor Assignment Problem. *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, 1921-1928.

Gavin, P. (2006). *Beginning Database Design*, Wiley Publishing, Inc., Indianapolis.

Glover F, Gary A, & Kochenberger E, (2003).*Handbook of Metaheuristics*, Kluwer Academic Publishers New York, Boston, Dordrecht, London, Moscow.

Goldberg, D.E., (1989), *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, Mass.: Addison-Wesley.

Hoffer J.A., Ramesh V., Topi H., (2011), *Modern Database Management, 10th Ed.*, Prentice Hall, New Jersey, pp. 61-63.

Jamaldeen F, (2019). Compare the structures of commercial and Islamic banks Available online at:, https://www.dummies.com/personal-finance/islamic-finance/compare-the-structures-of-commercial-and-islamic-banks. Accessed online on July 7, 2019.

Kirkpatrick S, Gelatt Jr. C.D. and Vecchi M.P. (1983). Optimization by Simulated Annealing, *Science*, 220, 671-680.

Lorena, L. A., Narciso, M. G., & Beasley, J. E. (2002). A Constructive Genetic Algorithm for the Generalized Assignment Problem. *Evolutionary Optimization, 5*, 1-19.

McCall, J. (2005). Genetic Algorithms for Modelling and Optimisation. *Journal of Computational and Applied Mathematics*, 184(1), 205–222.

Naveh Y., Richter Y., Connors D., & Altshuler Y. (2007). Workforce Optimization: Identification and Assignment of Professional Workers using Constraint Programming, *IBM Journal of Research and Development*, 51(3), 263 – 279.

Sahu A, Tapadar R, (2007). Solving the Assignment Problem using Genetic Algorithm and Simulated Annealing, *IAENG International Journal of Applied Mathematics*., 36(1), 1-7.

Stuart J. Russell and Peter Norvig, 1995, *Artificial Intelligence - A Modern Approach*, Prentice Hall, Englewood Cliffs, New Jersey.