Implementación de un algoritmo genético para la gestión y asignación de horarios para los colegios oficiales.

Juan Pablo Flórez Caicedo.

Cristian Camilo Giraldo Alvarez.

UNIVERSIDAD TECNOLÓGICA DE PEREIRA.

INGENIERÍA EN SISTEMAS Y COMPUTACIÓN.

PEREIRA – RISARALDA.

2018.

Implementación de un algoritmo genético para la gestión y asignación de horarios para los colegios oficiales.

Juan Pablo Flórez Caicedo.

Cristian Camilo Giraldo Alvarez.

Trabajo de grado, Proyecto de Aplicación.

Magister Ingeniero en sistemas Carlos Andrés López

UNIVERSIDAD TECNOLÓGICA DE PEREIRA.

INGENIERÍA EN SISTEMAS Y COMPUTACIÓN.

PEREIRA – RISARALDA.

2018.

Notas de Aceptación.

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Firma del Profesor Asignado.

_____

Firma de Jurado.

_____

Firma de Jurado.

DEDICATORIA.

A través de este largo camino que hemos recorrido estos últimos años, hemos tenido grandes alegrías, tristezas, sentimientos de agobio e incluso de dejarlo todo. Sin embargo nuestra propia persistencia, la ayuda de los que nos rodean y el deseo de progresar nos ha traído al último paso de este escalón que es nada más que el primero de los muchos escalones que nos esperan en el camino de ser mejores personas, ser profesionales íntegros, ser individuos que a través del conocimiento aporte aunque sea un grano de arena a esta sociedad tan golpeada por la desigualdad y la violencia.

Nosotros instamos a todos aquellos jóvenes que tengan la oportunidad de leer este documento que, no importa si no eres el más inteligente, o cuentas con muchos recursos o no, mientras perseveres y tengas una meta clara sin duda lograras cumplirla, mantén la cabeza en alto, respeta tus valores, mantén una mente abierta y no importa que te propongas lo lograras.

AGRADECIMIENTOS.

Agradecemos principalmente a nuestras familias que estuvieron allí cuando más los necesitamos nos brindaron su apoyo incondicional y nos guiaron en las etapas más tempranas de este proceso, al profesor “Carlos Andrés López” por guiarnos y ofrecernos consejo siempre sin intentar sesgarnos o influenciarnos directamente para así permitirnos lograr una solución propia, a la Universidad Tecnológica de Pereira por otorgarnos los espacios para realizar nuestra aplicación y las múltiples fuentes de información que tuvimos a disposición, por último y no menos importante a Dios por permitirnos vivir y poder experimentar todas las alegrías de la vida.

Contenido

1. **Glosario.**
   - Población Inicial: Entiéndase población inicial como los primeros valores que den solución al problema no necesariamente siendo una solución óptima.
   - Selección natural: Proceso que se produce en la naturaleza por el cual solo los individuos más fuertes o adaptados logran sobrevivir y reproducirse.

2. **Introducción.**

   Los algoritmos genéticos son una búsqueda basada en la teoría de la evolución de Darwin, estos algoritmos generalmente se basan en los sistemas naturales que debido a la constante evolución y perfeccionamiento permiten la solución de problemas de forma óptima.

   Estos algoritmos toman conceptos como la selección natural y la supervivencia del más apto. A través de la simulación de estos procesos naturales se pueden obtener resultados óptimos en problemas de alta complejidad, evolucionando las soluciones obtenidas hasta lograr valores óptimos del problema.

   La efectividad de estos algoritmos están ligados principalmente a la codificación de la población inicial ya que de esta depende si los métodos de evaluación tendrán suficiente flexibilidad, estos valores son los que a través de los métodos de evolución del algoritmo permitirán hallar una solución óptima al problema.

   Otros factores que afectan la efectividad de estos algoritmos son la población inicial, que no en todos los casos es un factor de mayor relevancia, pero en otros problemas podría afectar de forma negativa los resultados si el algoritmo no ha tenido en cuenta la posible aparición de óptimos locales, también la función objetivo que es la función que nos permite identificar que tan óptima es una solución.

   Con respecto a la implementación de los algoritmos genéticos se puede señalar que gracias a su relativa facilidad de concepto son fácilmente programables, pero esto depende también de la escala en la que se quiere construir el algoritmo, entiéndase como la escala las distintas posibilidades y factores agregados que pueden ser implementados adicionalmente en el algoritmo.

   Aunque los algoritmos genéticos son fáciles de implementar, su grado de efectividad les ha hecho merecedores de un gran interés en la comunidad de expertos, debido a que estos algoritmos abren la posibilidad de solucionar problemas muy complejos con relativamente pocos recursos y en periodo corto de tiempo.

   Aunque la efectividad de los algoritmos genéticos no puede ser cuestionada, surge la necesidad de tratar algunos problemas como los

mencionados anteriormente. También aumentar la calidad de los resultados obtenidos para obtener soluciones óptimas en cortos periodos de tiempo, Debido a estas necesidades surgen entonces los algoritmos genéticos modificados.

El concepto de Algoritmo genético modificado (AGM) fue introducido por primera vez por Michalewiez en 19992, en su trabajo propone la selección al azar de individuos para la reproducción e individuos destinados a morir. Estas selecciones teniendo en cuenta el valor de su función objetivo sirven para seleccionar los individuos con mayor valor para su posterior reproducción.

3. Antecedentes.

Los estudios sobre el desarrollo de sistemas evolutivos como medios para la optimización se remontan a los años 1950 y 1960. En 1957 Alex Fraser realiza varias publicaciones sobre la selección natural, dando inicio a la simulación por computadora de la evolución biológica, esto significo el primer paso en el establecimiento de muchos elementos que más tarde se convertirían en parte fundamental de los AG.

En 1960 Rechenberg propone un método para la optimización de parámetros de perfiles aerodinámicos en el documento “Evolution Strategies”, en 1966 Fogel, Owens y Walsh, crean el concepto de “Programación Evolutiva” y hacen una implementación de un algoritmo en una máquina de estados en la cual los diagramas de estados de transición evolucionaban mediante mutación aleatoria para encontrar la mejor solución.

Bagley menciona en 1967 por primera vez el término de algoritmos genéticos, aplicando este concepto en la búsqueda de conjuntos de parámetros en funciones de evaluación de juegos.

Pero es John Holland a quien se considera el creador de los AGs, gracias a la publicación de su libro “Adaptación en Sistemas Naturales y Artificiales” en 1975, Holland describe en su libro que los AGs son la abstracción de la evolución biológica, el método que usa en su libro es el desplazamiento de una población a otra nueva, usando el concepto de “selección natural” implementa los operadores genéticos como cruces, mutaciones e inversión.

A partir del año 1990 el avance en los algoritmos genéticos se evidencia en la cantidad de libros, publicaciones y conferencias que se dieron con base a este tema, esto permitió el flujo de información a las personas interesadas en el tema que poco a poco agregaron varias técnicas e ideas en los AGs que permitieron soluciones más óptimas.

En la actualidad los AGs son muy populares, abarcan un gran campo dentro de las ciencias como la ingeniería, la investigación de operaciones y más recientemente en la Inteligencia artificial, su popularidad es debido a que puede ser una técnica para abordar problemas para los cuales no existen técnicas o métodos especializados y sin embargo también puede dar muy buenas soluciones a los problemas que si cuentan con estas características.

4. Descripción del Problema.

La asignación de horarios en los colegios es una de las muchas acciones que se deben planear y gestionar antes de iniciar temporada escolar. Para esta actividad se deben tomar como restricciones varios atributos que definirán los horarios de clases para cada uno de los grados y subgrupos que tienen cada uno de los establecimientos académicos, estos atributos son:

Cantidad de docentes que pueden dar una asignatura, horas que deben cumplirse cada semana por asignatura, grados en el colegio y grupos por cada grado.

Cada colegio tiene n grados y por cada grado existen m grupos (ejemplo 4 sextos, 3 séptimos, 2 onces), a cada grado se asocian p asignaturas (ejemplo aritmética, castellano, etc), para cada asignatura se asocian q horas semanales (ejemplo aritmética 6 horas, ética 2 horas) además por ley se establece que el máximo número de horas por grado es de 30 semanales. Se establece que se tienen s docentes, por cada docente existen t asignaturas que un docente puede dictar (por ejemplo biología, química y ciencias naturales), por ley a un docente se le pueden asignar 22 horas semanales, sin embargo es posible asignar horas extras a un docente. Pueden aparecer restricciones de horarios y grupos.

Por lo tanto, las consideraciones que deben tomarse en cuenta en la asignación y gestión de horarios dependen del establecimiento académico, porque la cantidad de docentes, grados, grupos por cada grado e intensidad de las asignaturas dependen de cada colegio y sus modalidades de estudio, esto sugiere un problema organizacional para cada una de las instituciones educativas. Como solución a este problema se sugiere el uso de algoritmos que permitan obtener una asignación de horarios que satisfaga las necesidades de los centros educativos, algunos de estos algoritmos son los algoritmos genéticos.

5. **Objetivo General.**
   Implementar el algoritmo genético para la gestión y asignación de horarios en colegios oficiales.

6. **Objetivos Específicos.**

   - Establecer horarios que satisfagan las restricciones.
   - Diseñar el modelo de representación del problema.
   - Diseño del modelo de AG a implementar.
   - Entrenamiento del modelo (Validación cruzada).

7. **Metodología.**

   - **Establecer horarios que satisfagan las restricciones.**

     Como se menciona anteriormente en este documento, es necesario contar con una población inicial que cumpla los requisitos del problema, no tienen que ser soluciones óptimas, sin embargo es importante tener en cuenta que la selección de una población inicial óptima puede afectar el desempeño del algoritmo.

   - **Diseñar el modelo de representación del problema.**

     Este paso en el desarrollo es uno de los más importantes en el proyecto, ya que como se mencionaba antes, las respuestas que un AG pueda obtener de un problema están ligadas en gran medida a la forma en cómo se codifican los individuos de la población, en esta etapa se esperan lograr las siguientes metas:

     1. Un modelo de entradas que represente todas las posibles características de un individuo de la población.
     2. Un modelo de entradas que permita la mutación o evolución de los individuos.
     3. Un modelo de entradas que permita a los individuos de la población reproducirse.

- **Diseño del modelo de AG a implementar.**

  Los algoritmos genéticos tienen la propiedad de ser bastante libres en la forma en cómo se pueden implementar, pueden contar con variadas características, como un factor de aleatoriedad en la función de selección que en una baja probabilidad permita que individuos de la población no óptimos puedan reproducirse, esto con el fin de no saltarse posibles candidatos óptimos solo porque sus antecesores no cumplen con los valores mínimos.

  En esta etapa se decidirá qué propiedades debe tener el AG que será implementado, se espera plantear como mínimo:

  - ✓ La función objetivo: que permitirá saber que tan óptimo es un individuo de la población.
  - ✓ Función de selección: permitirá seleccionar los mejores individuos para su respectiva reproducción.
  - ✓ Función de reproducción: permite a dos individuos de la población reproducirse entre ellos.
  - ✓ Función de reemplazo: reemplaza un individuo de la población por otro más óptimo.
  - ✓ Función de descarte: elimina un individuo de la población que no cumpla con el criterio actual de optimalizad.

- **Costo Computacional de la solución.**

  El costo computacional se determinará una vez se tenga desarrollado el algoritmo, luego se evalúan las instrucciones para determinar una complejidad según su funcionalidad y sus restricciones.

  El costo computacional puede determinar la complejidad y los tiempos de ejecución que presenta el algoritmo, una vez obtenido estos datos sumándose con las pruebas de calidad del modelo, se puede determinar la eficacia del modelo, poniendo dicho modelo en categorías, como viable, no tan viable o no es viable el uso del algoritmo.

- **Entrenamiento del modelo**

  El entrenamiento del modelo se hará usando el método de validación cruzada hold-out, usando primeramente un 70% y luego un 90% de los datos de entrenamiento, supervisando las salidas del algoritmo.

  La validación cruzada hold-out consiste en usar una parte del conjunto de datos de entrenamiento, esto se hace para obtener

mejor respuesta en el algoritmo puesto que usar todos los datos de entrenamiento desencadena que el algoritmo converge en soluciones idénticas y no encuentre soluciones que puedan tener mejores resultados.

- **Prueba de calidad del modelo**

  Las pruebas de calidad del modelo son con la matriz de confusión, donde se determinara la cantidad de positivos ciertos, positivos falsos, negativos ciertos y negativos falsos, que darán un mejor entendimiento de las fortalezas y debilidades del algoritmo.

  Esta prueba de calidad, ayudará a determinar el margen de error que tiene el algoritmo genético aplicado.

8. Desarrollo.

1. Diseño del modelo de representación del problema.

1.1. Estudio de las variables que interfieren en el problema.

Durante el análisis se identificaron un conjunto extenso de variables que afectan la solución de la problemática que representa la generación automática de horarios para un colegio oficial, tales como, profesores, límite de horas semanales laborales del profesor, límite de horas a la semana de una materia por grado, salones, grados, asignaturas, intensidad horaria, días, horas.

Sin embargo, una solución que implique todas las variables mencionadas anteriormente aumenta la complejidad del desarrollo, por lo tanto, procedimos a seleccionar un conjunto de variables pequeño que nos permitió describir el problema e implementarlo de forma más sencilla:

Grados, esta variable contiene cada uno de los grados que instruye el colegio.

Asignaturas, esta variable contiene todas las asignaturas instruidas en el colegio.

Intensidad horaria, esta variable es la conexión entre las asignaturas y los grados que representa la cantidad de horas académicas a la semana.

Días, contiene un conjunto de horas, en este caso con un máximo de 6 horas por día.

Horas, es la mínima expresión de las variables del sistema, esta contiene la relación asignatura, grado e intensidad horaria.

1.2. Análisis de las posibles codificaciones de las variables del problema.

En el desarrollo de la solución, pudimos identificar dos tipos de estructuras de datos para la codificación de las variables.

Usando listas, encontramos una posible codificación usando listas enlazadas que a través de las clases Horas y Materias representaban toda la asignación horaria por cada uno de los grupos. Sin embargo se presentaron dificultades a la hora de representar la dependencia de una asignatura respecto a cada uno de los grados, esto provocaba un aumento en la complejidad del uso de las listas enlazadas.

Usando Matriz de intersección, encontramos una solución con menos costos computacionales y algorítmicamente mejor, esta solución nos permitió representar de forma única los atributos de un salón (considerado uno de los atributos las materias que ve cada salón).

1.3. Codificación de las variables de representación del problema.

La codificación de las variables que representan el problema se encuentra en el código debidamente comentadas, sin embargo, la solución al problema se planteó de la siguiente manera:

Generador

1. Introducir en la base de datos NoSQL, las entradas que son una matriz de grados vs materias (donde los datos son las horas de intensidad semanales por grado)

Grados	1°	2°	3°	4°	5°	6°
Matematicas	5	5	5	5	5	4
Español	5	5	5	5	5	4
Religion	2	2	2	2	2	2
Etica	1	1	1	1	1	1
Fisica	0	0	0	0	0	3
Geometria	2	2	2	2	2	0
Quimica	0	0	0	0	0	0

Tabla 1 Asignaturas vs grados, Elaboración propia

Esto genera una colección, cuya estructura es:

```
"Nombre Materia ": Matemáticas,
"Grados": [1°, 2°, 3°, 4°, 5°, 6°],
"Intensidades": [5, 5, 5, 5, 5, 4]
```

"**Nombre Materia**", es el atributo que contiene el nombre de la materia que se está ingresando.
"**Grados**", es una lista de nombres de cada uno de los grados del colegio.
"**Intensidades**", es una lista de horas donde se encuentra en cada una de las posiciones de la lista la cantidad de horas semanales por cada uno de los grados en el colegio, si un grado no dicta la materia, deberá aparecer un 0 en la posición correspondiente al grado.

2. Generamos horarios utilizando la colección del anterior punto, primero transformamos cada materia en un token, este token contiene el nombre de la asignatura y su intensidad de horas semanales para un grado.
   Obteniendo la siguiente estructura:

```
"nombreMateria": "matemáticas"
"intensidad": 5
```

3. Una vez obtenidos los tokens para un grado, se repite el nombre de cada una de las materias en una lista según la intensidad que posee cada una de las asignaturas, con el fin de obtener una lista de 30 elementos, los cuales corresponden a las 6 horas diarias de 5 días a la semana.
4. Luego procedemos a realizar una selección aleatoria en la lista de 30 elementos, para generar un día de 6 horas, cada elemento seleccionado es sacado de la lista y continuamos seleccionando aleatoriamente. Al completar un día seguimos con el siguiente, hasta completar 5 días que corresponden a lunes, martes, miércoles, jueves y viernes.
5. Terminado el día viernes, se puede generar una nueva estructura para ser ingresada a la base de datos, la estructura es la siguiente:

```
"Grado": "6",

"lunes": ["castellano", "sociales",
"matemáticas", "religión", "sociales",
"castellano"],

"martes": ["castellano", "educación física",
"castellano", "tecnología", "biología",
"sociales"],
```

```
"miércoles": ["biología", "matemáticas",
"artes", "matemáticas", "castellano",
"ingles"],

"jueves": ["matemáticas", "biología",
"matemáticas", "ingles", "dibujo",
"educación física"],

"viernes": ["artes", "ética", "sociales",
"biología", "ingles", "tecnología"]
```

“Grado”, atributo encargado de almacenar el grado o salón al cual va ser asignado el horario generado.
“lunes”, lista de asignaturas, la cual contiene las asignaturas asignadas al día lunes.
“martes”, lista de asignaturas, la cual contiene las asignaturas asignadas al día martes.
“miércoles”, lista de asignaturas, la cual contiene las asignaturas asignadas al día miércoles.
“jueves”, lista de asignaturas, la cual contiene las asignaturas asignadas al día jueves.
“viernes”, lista de asignaturas, la cual contiene las asignaturas asignadas al día viernes.

6. Por cada uno de los grados se debe seguir los procesos de los literales 2,3,4 y 5. Para este caso de estudio se usaron 1000 sujetos (horarios) por cada uno de los grupos en la primera generación.

1.4. Algoritmo Genético.

Desde un punto de vista general las técnicas de Computación Evolutiva, como los Algoritmos Genéticos o la Programación Genética, pueden considerarse como un conjunto de técnicas computacionales más ligadas en sus conceptos a los procesos biológicos que a las técnicas computacionales tradicionales (Gestal, 2018).

La mayoría de los algoritmos genéticos comparten una estructura general, lo que les permite ser programados fácilmente, sin embargo esto no los convierte en algoritmos inflexibles, esta estructura puede estar sujeta a cambios, bien sea para mejorar el desempeño o simplemente para diversificar las respuestas del algoritmo.

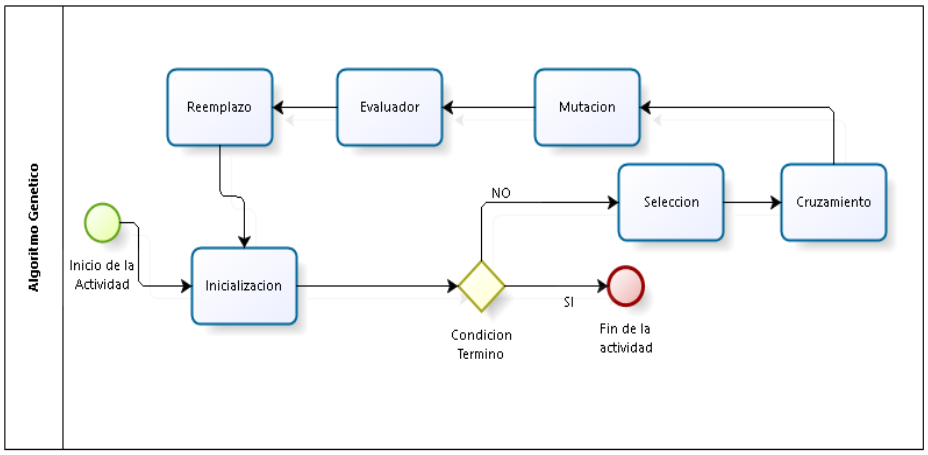


Ilustración 1, Diagrama de Procesos AGs

Como se puede observar en la ilustración 1, la estructura del algoritmo genético no es difícil de entender ni contiene demasiados procesos. Se debe tener en cuenta que este diseño de algoritmo genético no contempla el formato de los genes de la población, por lo tanto la codificación de individuos y la recombinación de estos quedan al criterio del desarrollador.

1.5. Diseño de Algoritmo genético a implementar.

Siguiendo la estructura general previamente expuesta desarrollamos un algoritmo con una estructura estándar que nos permitió integrarlo fácilmente en la ejecución del programa:

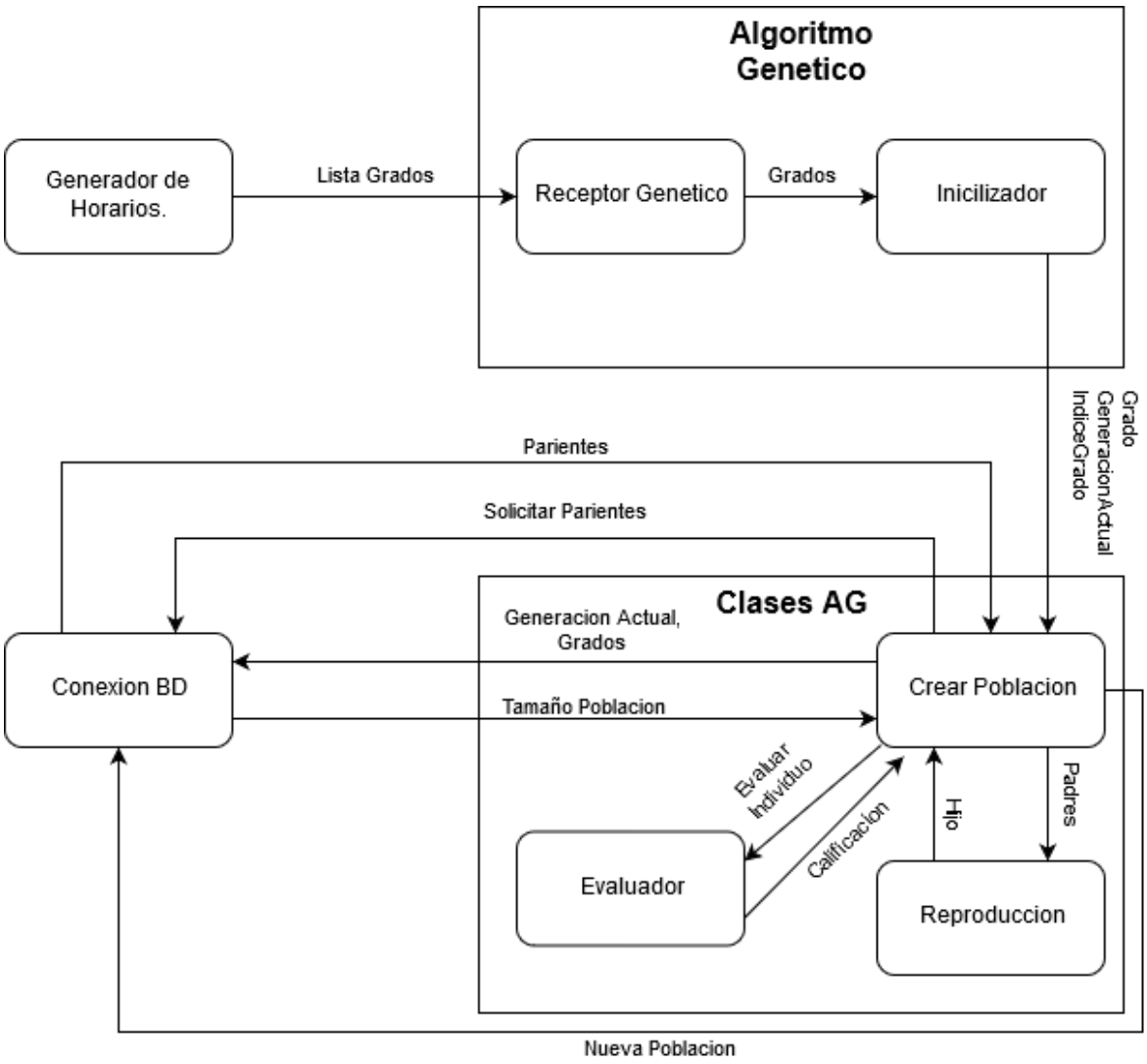


Ilustración 2 Grafico de la estructura del AG

Este grafico nos permite vislumbrar la estructura del algoritmo genético que implementamos en el proyecto, en el podemos identificar las relaciones entre cada uno de los componentes de la implementación, las funciones críticas y darnos una idea del flujo de datos.

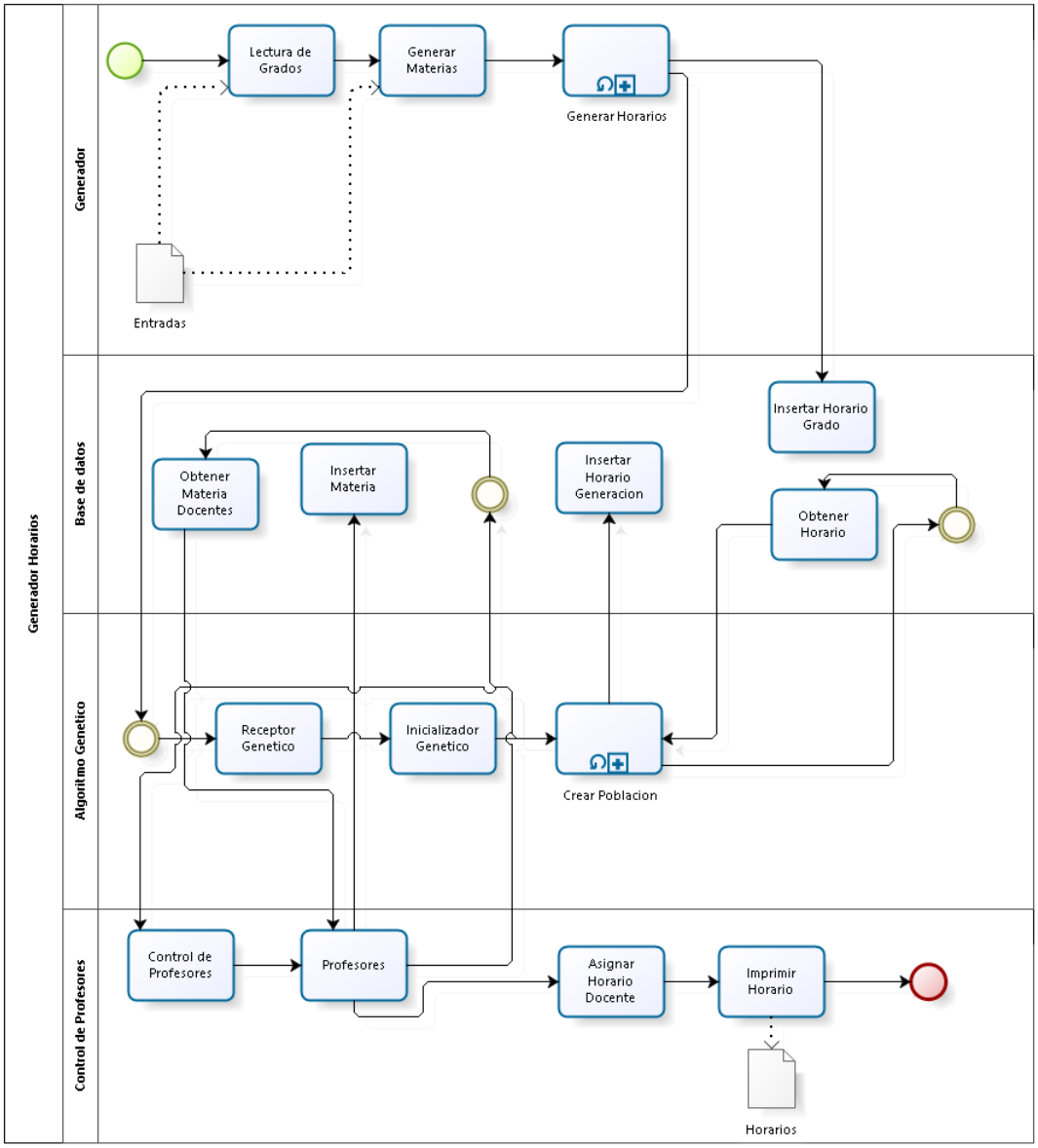
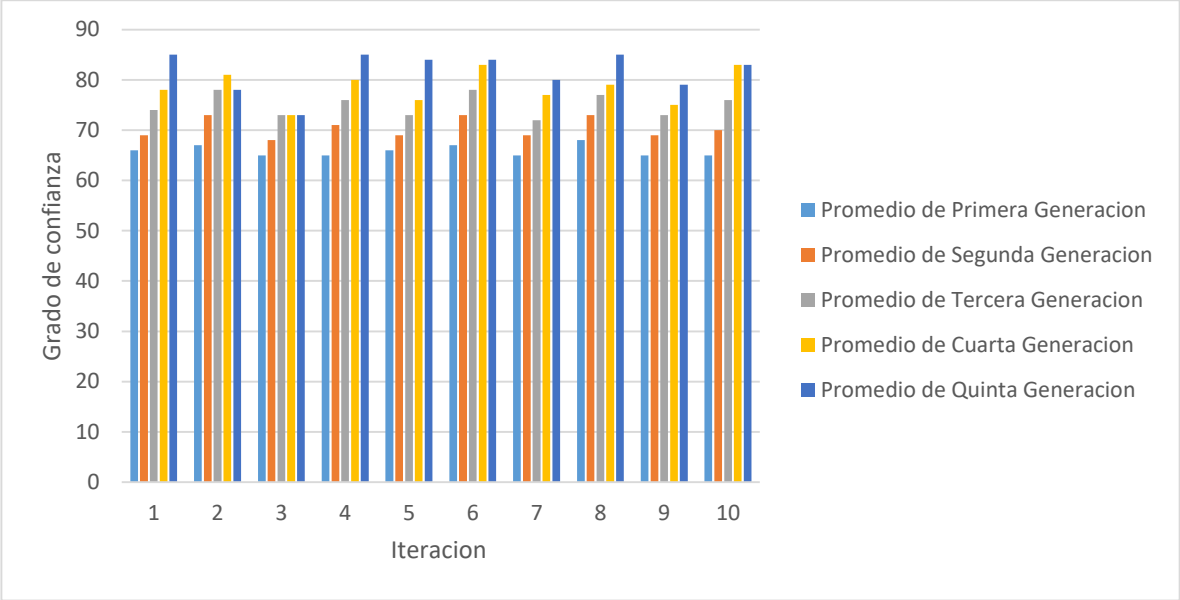


Ilustración 3, Diagrama de Procesos AG Implementado.

En la Ilustración 3 se aprecia el diagrama de procesos del algoritmo genético y su integración con los distintos módulos del aplicativo que incluyen la comunicación con la base de datos, una vista general de cómo cada proceso de los módulos que componen el aplicativo interactúan durante la ejecución.

1.6. Evaluación del modelo implementado.



En esta grafica se observa la iteración vs el grado de confianza promedio, permitiendo observar una clara tendencia a la mejora de los individuos de la población en cada generación teniendo como resultado un 70% de mejora en cada 10 iteraciones.

1.7. Marco Teórico.

- **NoSQL**

  Este término surge en 1998, por Carlo Strozzi para referirse a una base de datos relacional de código abierto en el que no se usa el lenguaje de consultas SQL, luego fue retomado en el año 2009 por Johan Oskarsson quien organizo un evento para tratar las bases de datos distribuidas de código abierto no relacionales (Rguez, 2012). Las bases de datos NoSQL (no solo SQL), crecieron con las principales redes sociales, donde Google, Amazon, Twitter y Facebook, tenían que enfrentarse a menudo con el tratamiento de los datos en las que las bases de datos tradicionales no encontraban solución, además con el crecimiento de la web en tiempo real, existió la necesidad de procesar grandes volúmenes de información por lo que las NoSQL son una respuesta a los complejos datos que van apareciendo en la actualidad, ampliando el sistema relacional a un enfoque más flexible,

pero más enfocado al uso de la información este retira el uso de Joins (productos cartesianos entre tablas) y normalizaciones para optar por una alternativa en la que cada registro tiene su propia información, logrando que en una misma colección exista diversidad de campos y estructuras entre los registros (Andres Araujo, 2016).

Las bases de datos NoSQL son una herramienta importante en la actualidad para el uso de información en tiempo real, brindan la capacidad de almacenamiento dinámico, además de agilidad en las consultas de la misma base de datos (este caso es para las bases de datos NoSQL que son más maduras como MongoDB). Cabe resaltar que las bases de datos NoSQL no garantizan las propiedades ACID (atomicidad, consistencia, aislamiento, durabilidad) que son unos parámetros que permiten clasificar las transacciones de los SGBD (Sistemas de gestión de bases de datos).

- **MongoDB.**

  Es un sistema de bases de datos NoSQL cuyo desarrollo empezó en octubre de 2007 por la compañía software 10gen, este es desarrollado en lenguaje C++ lo que lo confiere cierta cercanía al *bare metal*, o recursos de hardware de la máquina, de modo que es bastante rápido a la hora de ejecutar sus tareas. (paramio, 2011) Además esta licenciado como GNU AGPL 3.0, que lo convierte en un software de licencia libre.

  Su principal estructura se denomina documento, estos pueden agrupar colecciones que son equivalentes a las tablas en las bases de datos SQL, con la gran diferencia de que cada registro puede contener diferentes atributos en una misma tabla estos documentos son guardados en formato BSON que son una representación binaria de JSON, este tipo de formato permite realizar búsquedas rápidas de datos sin importar que su tamaño de almacenamiento aumente, puesto que se considera que el hardware de almacenamiento es menos costoso, por lo que es más importante la velocidad de localización de información que el coste de almacenamiento.

- **Python**

  Python es un lenguaje de programación creado a finales de los 80 por Guido Van Rossum, en el centro de matemáticas y la informática (CWI,Centrum Wiskunde & informática), este es un lenguaje de programación

poderoso y fácil de aprender enfocado a la programación orientada a objetos. Su sintaxis y tipado dinámico, permiten realizar desarrollos rápidos de aplicaciones en diversas áreas, además cuenta con muy buena documentación y muchos módulos libres desarrollados por terceros.

Python cuenta con un conector sencillo a mongoDB, además el uso de listas es uno de sus principales fuertes, por lo que Python es un excelente lenguaje de programación para este tipo de desarrollos.

1.8. Conclusiones.

- Debido a la naturaleza aleatoria del algoritmo, algunas veces se presentan casos donde una generación es ligeramente inferior a su predecesora.
- Se presentan casos en los que el grado de confianza promedio se mantiene en 2 o más generaciones, lo que nos sugiere que se presentan máximos locales que hacen que la recombinación de los individuos no cambie demasiado su grado de confianza, sospechamos que esto sucede cuando el resultado de la prueba entre padres e hijo, el ganador es siempre el mismo individuo.
- Proponemos implementar medidas en contra de estos máximos locales, una posible mitigación de este problema es mediante la integración de individuos mutados (no necesariamente óptimos) dentro de cada población, aumentando la diversidad de los individuos.
- No se observa la necesidad de aumentar la población inicial del programa, con 6000 individuos se obtienen horarios con un puntaje (grado de confianza) promedio por encima de 70 puntos lo que los hace alternativas bastante viables.

1.9. Bibliografía y Web grafía.

- http://eddyalfaro.galeon.com/geneticos.html.
- http://repositorio.utp.edu.co/dspace/bitstream/handle/11059/5384/0063823A696.pdf?sequence=1
- https://books.google.com.co/books?id=0eznlz0TF-IC&lpg=PA2&ots=shiC5W35Lf&dq=Rechenberg%201960&pg=PA2#v=onepage&q=Rechenberg%201960&f=false
- https://revistadigital.inesem.es/informatica-y-tics/bases-datos-nosql-mongodb/
- https://riunet.upv.es/bitstream/handle/10251/80558/4153-15282-1-PB.pdf?sequence=1

- https://www.genbetadev.com/bases-de-datos/el-concepto-nosql-o-como-almacenar-tus-datos-en-una-base-de-datos-no-relacional
- https://revista.jovenclub.cu/bases-de-datos-nosql/
- https://www.genbetadev.com/bases-de-datos/mongodb-que-es-como-funciona-y-cuando-podemos-usarlo-o-no
- Mongodb the defitive guide – Kristina Chodorow 1401.