PROGRAMACION DISTRIBUIDA DE HORARIOS EN JAVA UTILIZANDO TECNICAS DE ALGORITMOS GENETICOS

GERARDO ALFONSO VERJEL CLAVIJO

UNIVERSIDAD AUTONOMA DE BUCARAMANGA
FACULTAD DE INGENIERÍA DE SISTEMAS
LINEA DE INVESTIGACIÓN EN SISTEMAS DE INFORMACION E INGENIERIA DEL SOFTWARE
BUCARAMANGA
2005

PROGRAMACION DISTRIBUIDA DE HORARIOS EN JAVA UTILIZANDO TECNICAS DE ALGORITMOS GENETICOS

GERARDO ALFONSO VERJEL CLAVIJO

Tesis de Grado para optar el titulo de
Ingeniero de Sistemas

Director
Ing. EDUARDO RINCON SERRANO

UNIVERSIDAD AUTONOMA DE BUCARAMANGA
FACULTAD DE INGENIERÍA DE SISTEMAS
LINEA DE INVESTIGACIÓN EN SISTEMAS DE INFORMACION E INGENIERIA DEL SOFTWARE
BUCARAMANGA
2005

Nota de aceptación

_____
_____
_____

_____

Presidente del Jurado

_____

Evaluador

_____

Evaluador

Bucaramanga, 22 de agosto de 2005

DEDICATORIA

Mis papás, porque me dieron la vida y entregaron la suya por verme crecer y lograr mis sueños, que un día se convirtieron en nuestros sueños gracias a la grandeza de su amor.

Mis Hermanas y mi sobrino, porque en ellos he encontrado los mejores amigos para compartir cada sonrisa y cada lágrima que surgen cuando estas creciendo y aprendiendo a vivir.

TABLA DE CONTENIDO

INTRODUCCIÓN

1. OBJETIVOS 15

1.1 OBJETIVO GENERAL 15

1.2 OBJETIVOS ESPECÍFICOS 15

2. ALGORITMOS GENETICOS 16

2.1 COMO TRABAJA UN ALGORITMO GENÉTICO 17

2.1.1 Codificación del Dominio 17

2.1.2 Evaluación de la Población 18

2.1.3 Función de Adaptación 20

2.1.4 Selección 21

2.1.5 Cruce 22

2.1.6 Mutación 23

3. El ALGORITMO GENETICO SIMPLE (AGS) 24

4. ALGORITMOS GENETICOS PARALELOS 26

4.1 DIFERENCIA ENTRE EL ALGORITMO GENÉTICO SIMPLE Y EL ALGORITMO GENÉTICO PARALELO 27

4.2 DIVISIÓN DE LOS ALGORITMOS GENÉTICOS PARALELOS 28

4.2.1 Algoritmo Genético Paralelo global o de una sola población 28

4.2.2 Algoritmos Genéticos de Múltiples poblaciones o Distribuido 30

4.2.3 Diferencias entre El Algoritmo Genético Global o de una solo población y el Algoritmo Genético Paralelo Distribuido o de múltiple población. 32

5. PROGRAMACION DISTRIBUIDA 33

5.1 SERVICIO DE BÚSQUEDA 34

5.2 RMI (Remote Method Invocation) 34

5.3 JDBC 35

6. DESARROLLO DEL PROYECTO 36

6.1 ANALISIS Y DISEÑO 36

6.1.1 Declaración del Problema 36

6.2 ANÁLISIS DE REQUERIMIENTOS DEL SISTEMA DE PROGRAMACIÓN DE HORARIOS 38

6.2.1 Requerimientos Funcionales del Sistema de Horarios 38

6.2.2 Requerimientos no funcionales 40

6.2.3 Seudorequerimientos 41

6.2.4 Escenarios 43

6.3 CASOS DE USO DEL SISTEMA DE PROGRAMACIÓN DE HORARIOS. 45

6.3.1 Actores. 45

6.3.2 Casos de Uso 45

6.3.3 Diagrama de Casos de Uso Modulo de Generación de Horarios 48

6.3.4 Diagramas de Secuencia del sistema de Horarios. 48

6.3.4.1 Diagrama de Secuencia del caso de uso Ingresar Datos 49

6.3.4.1.1 Diagrama de Colaboración del caso de uso Ingresar Datos 50

6.3.4.2 Diagrama de Secuencia del caso de uso Buscar Datos 51

6.3.4.2.1 Diagrama de Colaboración del caso de uso Buscar Datos 52

6.3.4.3 Diagrama de Secuencia del caso de uso Eliminar Datos 53

6.3.4.3.1 Diagrama de Colaboración del caso de uso Eliminar Datos 54
6.3.4.4 Diagrama de Secuencia del caso de uso Actualizar Datos 55
6.3.4.4.1 Diagrama de Colaboración del caso de uso Actualizar Datos 56
6.3.4.5 Diagrama de Secuencia del caso de uso Generar Horarios 57
6.3.4.5.1 Diagrama de Colaboración del caso de uso Generar Horarios 58
6.3.4.6 Diagrama de Secuencia del caso de uso Consultar Horarios 59
6.3.4.6.1 Diagrama de Colaboración del caso de uso Consultar Horarios 60
6.3.4.7 Diagrama de Secuencia del caso de uso Modificar Horarios 61
6.3.4.7.1 Diagrama de Colaboración del caso de uso Modificar Horarios 62
7. DICCIONARIO DE DATOS 63
8. DESARROLLO DEL ALGORITMO GENETICO 84
8.1 POBLACIÓN INICIAL 84
8.2 FUNCION DE EVALUACION 86
8.3 OPERADORES GENETICOS 90
8.3.1 Operador de Selección 90
8.3.2 Operador de Cruce 91
8.3.3 Operador de mutación 93
9. CONCLUSIONES 96
REFERECIAS

LISTA DE FIGURAS

FIGURA 1 Algoritmo genético paralelo Master – Esclavo 30

FIGURA 2 Algoritmo genético paralelo con población especial distribuida 31

FIGURA 3 Algoritmo genético paralelo Múltiple – Población 31

FIGURA 4 Diagrama de Casos de Uso Generar Horarios 48

FIGURA 5 Diagrama de Secuencia Ingresar Datos 49

FIGURA 6 Diagrama de Colaboración Ingresar datos 50

FIGURA 7 Diagrama de Secuencia Buscar Datos 51

FIGURA 8 Diagrama de Colaboración Buscar Datos 52

FIGURA 9 Diagrama de Secuencia Eliminar Datos 53

FIGURA 10 Diagrama de Colaboración Eliminar Datos 54

FIGURA 11 Diagrama de Secuencia Actualizar Datos 55

FIGURA 12 Diagrama de Colaboración Actualizar Datos 56

FIGURA 13 Diagrama de Secuencia Genera Horarios 57

FIGURA 14 Diagrama de Colaboración Generar Horarios 58

FIGURA 15 Diagrama de Secuencia Consultar Horarios 59

FIGURA 16 Diagrama de Colaboración Consultar Horarios 60

FIGURA 17 Diagrama de Secuencia Modificar Horarios 61

FIGURA 18 Diagrama de Colaboración Modificar Horarios 62

FIGURA 19 Tabla de Evaluación 89

FIGURA 20 Selección de Horarios 91

FIGURA 21 Cruce de Horarios 93

FIGURA 22 Mutación de Horarios 95

RESUMEN

La Universidad Autónoma de Bucaramanga requiere un modulo de software que facilite la automatización de la elaboración de los horarios de las asignaturas de los diferentes programas académicos de la universidad. La programación de los horarios de las asignaturas de cada una de las facultades que conforman la Universidad es un problema multivariable que involucra optimización de recursos. Entre las variables determinantes se encuentran la cantidad de docentes, su disponibilidad horaria, las asignaturas que pueden impartir; la cantidad, capacidad y disponibilidad horaria de los salones; entre otras. Además existen ciertas restricciones que complican aún más esta tarea como por ejemplo: las franjas horarias según los niveles, evitar los cruces de asignaturas que pertenecen a un mismo nivel, evitar los cruces de los horarios de los docentes, etc.

Esto en la actualidad se hace con ayuda de herramientas ofimáticas como Word y Excel, causando inconsistencias al momento de integrar toda la información.

El desarrollo de un algoritmo tradicional para la solución de este tipo de problema resultaría bastante complejo, por lo que se utilizaron técnicas de algoritmos genéticos que simulan el proceso de evolución natural.

El objetivo principal del sistema de Programación de Horarios en Java Utilizando Técnicas de algoritmos Genéticos es la realización de los horarios de los docentes de la universidad Autónoma de Bucaramanga con el fin de evitar el cruce de Docentes, Salones, Cursos de un mismo nivel, Cursos que sean requisitos de otros Cursos, respetar la disponibilidad de los docentes entre otros.

El Sistema trabaja con la informacion suministrada por los jefes de Departamento o los Coordinadores de Facultad, esta información es la siguinete:

Disponibilidad horaria de los docentes a su cargo: Si el docente es de tiempo completo su disponibilidad corresponde a la totalidad de horas estipuladas en su jornada laboral y aquellas horas por fuera de su horario normal siempre y cuando se cuente con la aprobación del docente. Si el docente es de hora cátedra su disponibilidad no debe ser inferior al total de horas de la carga asignada para el semestre a programar.

Carga académica de cada docente: Indica cuantas secciones de cada curso corresponden a cada docente y se elabora con base en la demanda de cursos proyectada para el semestre a programar. La demanda puede establecerse mediante consultas al sistema BANNER y datos estadísticos acerca del ingreso de estudiantes nuevos durante los últimos semestres.

Horarios Preestablecidos: Indica aquellos horarios de reunión que el sistema debe respetar ya que no pueden ser modificados por diversas circunstancias como disponibilidad del docente, recursos, etc.

Adicionalmente se debe extraer del sistema BANNER la siguiente información:

Datos de los docentes: Información básica como el ID, Apellidos, Nombres, Dedicación, Dependencia.

Datos de los salones: Código, Campus, Edificio, Salón, Capacidad, Tipo de Salón (dependiendo de los recursos con que cuenta).

Información referente a los planes de estudios de los diferentes programas. Todos los salones sin excepción tendrán una disponibilidad de 6:00 AM a 10:00 PM de lunes a domingo.

El sistema genera los horarios de los docentes utilizando Algoritmos geneticos, ya que estos algoritmos simulan el proceso de evolucion natural con el fin de encontrar la mejor solución al problema planteado. El sistema utiliza los operadores de Selección, Cruce y Mutacion con el fin de encontrar las soluciones mas óptima que se adapte al problema de la programación de Horarios.

El sistema es administrado por los jefes de departamento o coordinadores de área de cada Facultad de la Universidad Autónoma de Bucaramanga, ya que ellos son los encargados de la programación de los horarios de cada uno de los docentes de la universidad.

En primer lugar el sistema es puesto en funcionamiento con la escuela de ingenierias de la universidad, después sera utilizado por todas las facultades que conforman la Universidad Autonoma de Bucaramanga.

INTRODUCCION

En la actualidad la universidad no cuenta con un modulo que optimice la programación de los horarios de los diferentes cursos, ya que esta tarea se hace manualmente o con la ayuda de herramientas ofimáticas como Word y Excel causando traumatismo a la hora de integrar la información.

Para solucionar este problema se desarrollo un software que realice la optimización de los horarios.

Para el desarrollo de este sistema se utilizaron técnicas de algoritmos genéticos ya que estos simulan el proceso de evolución natural y utilizar algoritmos tradicionales para realizar esta tarea resultaría bastante complejo de solucionar.

El sistema de programación de horarios plantea la solución mas óptima ha este problema pues los Algoritmos Genéticos buscar la mejor solución a los problema planteados, cabe anotar que estos no encuentran una solución completa al problema, estos Algoritmos tratan de encontrar la mejor solución que sea posible.

1. OBJETIVOS

1.1 OBJETIVO GENERAL

Desarrollar un software que permita optimizar la programación de los horarios de cada una de las asignaturas de las diferentes facultades que conforman la Universidad Autónoma de Bucaramanga utilizando técnicas de algoritmos genéticos.

1.2 OBJETIVOS ESPECÍFICOS

Identificar el dominio del problema y codificar el espacio de soluciones para obtener estructuras manejables para así poder trabajar con algoritmos genéticos, así poder encontrar la mejor solución en esta población.

Diseñar e implementar los operadores genéticos de selección, cruce y mutación que permitan aplicar técnicas de algoritmos genéticos a la solución del problema de generación de horarios de la UNAB.

Seleccionar y aplicar el patrón o patrones de diseño que permitan el procesamiento distribuido de la información.

Utilizar la herramienta para poder trabajar con Programación Distribuida con el fin de poder utilizar al máximo el rendimiento del sistema.

Modelar en UML cada una de las fases del ciclo del desarrollo del Software.

2 ALGORITMOS GENETICOS

Los Algoritmos Genéticos (AG) son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización. Están basados en el proceso genético de los organismos vivos. A lo largo de las generaciones, las poblaciones evolucionan en la naturaleza de acorde con los principios de la selección natural y la supervivencia de los más fuertes. Por imitación de este proceso, los Algoritmos Genéticos son capaces de ir creando soluciones para problemas del mundo real. La evolución de dichas soluciones hacia valores óptimos del problema depende en buena medida de una adecuada codificación de las mismas.

Un algoritmo genético consiste en una función matemática o una rutina de software que toma como entradas a los ejemplares y retorna como salidas cuales de ellos deben generar descendencia para la nueva generación.

Versiones más complejas de algoritmos genéticos generan un ciclo iterativo que directamente toma a la especie y crea una nueva generación que reemplaza a la antigua una cantidad de veces determinada por su propio diseño.

Los Algoritmos Genéticos usan una analogía directa con el comportamiento natural. Trabajan con una población de individuos, cada uno de los cuales representa una solución factible a un problema dado. A cada individuo se le asigna un valor ó puntuación, relacionado con la bondad de dicha solución. En la naturaleza esto equivaldría al grado de efectividad de un organismo para competir por unos determinados recursos. Cuanto mayor sea la adaptación de un individuo al problema, mayor será la probabilidad de que el mismo sea seleccionado para

reproducirse, cruzando su material genético con otro individuo seleccionado de igual forma. [1]

2.1 COMO TRABAJA UN ALGORITMO GENÉTICO

Los Algoritmos genéticos simulan el proceso de evolución natural para cualquier problema a resolver, ellos necesitan conocer como codificar el dominio de la población, necesitan tener una función de evaluación con el fin de encontrar a los mejores individuos de cada población y también debe trabajar con los operadores de selección cruce y mutación.

2.1.1 Codificación del Dominio

Para un algoritmo genético lo primero que se requiere es determinar en qué espacio se encuentran las posibles soluciones al problema que se pretende resolver. En caso de tener un problema de optimización de una función cuyo dominio es un subconjunto de los números reales, entonces este subconjunto es al que nos referimos. Pero el algoritmo opera sobre “códigos genéticos”, sobre genotipos que se deberán mapear al espacio de soluciones. Es decir, es necesario codificar de alguna manera el dominio del problema para obtener estructuras manejables que puedan ser manipuladas por el AG. Cada una de estas estructuras constituye el equivalente al genotipo de un individuo en términos biológicos. El elemento del dominio del problema al que se mapea este genotipo es el análogo al fenotipo. Es frecuente que el código de los elementos del dominio del problema utilice un alfabeto binario (0’s y 1’s). Una vez que se ha definido la manera de codificar los elementos del dominio del problema y se conoce la forma de pasar de un elemento a su código y viceversa, es necesario fijar un punto de partida. Los algoritmos genéticos manipulan conjuntos de códigos en

[1] http://www.ica.ele.puc-rio.br/pesquisa/download/paper.pdf

generaciones sucesivas. Nuevamente haciendo una analogía, manipulan poblaciones de códigos. En éstas un código puede aparecer más de una vez. El algoritmo se encargará de favorecer la aparición en la población de códigos que correspondan a elementos del dominio que estén próximos a resolver el problema.

El algoritmo recibirá como entrada una población de códigos y a partir de ésta generará nuevas poblaciones, donde algunos códigos desaparecerán mientras que otros, que se mapean en mejores soluciones posibles, aparecen con más frecuencia hasta que se encuentra una satisfactoria o hasta que se cumple alguna otra condición de terminación. Los códigos en una población, es decir, los elementos de ésta serán llamados individuos y a los códigos en general, ya no en el contexto exclusivo de una población, se les denominará cromosomas, genotipo, genoma o código genético, por analogía con los términos biológicos de donde surgen. [2]

2.1.2 Evaluación de la Población.

En la naturaleza hay individuos más hábiles que otros para sobrevivir. En una manada de gacelas hay unas más rápidas que otras, hay algunas enfermas o propensas a enfermar, hay algunas más débiles que otras. Todas las características mencionadas señalan alguna diferencia entre los individuos. Además, esta diferencia es relativa, es decir, siempre está referida al resto de la población de gacelas. Se dice: “ésta es más rápida que el resto de la población” o “aquélla es más saludable que el promedio de sus congéneres”. De alguna manera, siempre se relaciona al individuo con la población a la que pertenece. Si se considera cada una de estas características como medidas del desempeño de cada individuo, se está hablando de que el desempeño de cada individuo de la población está en función del desempeño de sus congéneres. Por ejemplo, ¿qué

[2] http://www.ica.ele.puc-rio.br/pesquisa/download/paper.pdf

tan veloz debe ser una gacela para evitar ser cazada por un cazador? Si es más rápida que el cazador está salvada. Pero lograr eso es difícil, casi no habría gacelas. Más bien la respuesta correcta es relacionar al individuo con el resto de su población y decir: debe ser más rápida que la gacela más lenta, de este modo el depredador perseguirá a otra (claro que después de la primera cacería la medida de desempeño cambia porque lo ha hecho la población misma). Al igual que en la naturaleza, en los algoritmos genéticos es necesario establecer algún criterio que permita decidir cuáles de las soluciones propuestas en una población son mejores respecto del resto de las propuestas y cuáles no lo son. Es necesario establecer, para cada individuo, una medida de desempeño relativa a la población a la que pertenece. Para determinar cuáles de estos individuos corresponden a buenas propuestas de solución y cuáles no, es necesario calificarlos de alguna manera. Cada individuo de cada generación de un algoritmo genético recibe una calificación o, para usar el término biológico, una medida de su grado de adaptación. Éste es un número real no negativo tanto más grande cuanto mejor sea la solución propuesta por dicho individuo. El objetivo de este número es que permita distinguir propuestas de solución buenas de aquéllas que no lo son. Si el problema a resolver consiste en maximizar una función, entonces la calificación asignada a un individuo determinado debe indicar qué tan alto es el valor de la función en el elemento de su dominio codificado por el individuo. Si, en cambio, el problema es determinar la ruta más corta entre dos puntos, la calificación deberá ser tanto más alta cuanto más corto sea el camino codificado en el individuo que esté siendo calificado. Evidentemente, al hablar de que a cada individuo de la población se le asigna una y sólo una calificación, se está hablando de una función que se denomina función de adaptación, cuya evaluación puede no ser sencilla y es, de hecho, lo que en la mayoría de los casos consume más tiempo en la ejecución de un algoritmo genético. Hay que tener en cuenta que se evalúa una vez en cada individuo de cada generación. Si un AG es ejecutado con una población de tamaño 100 durante 100 generaciones, la función es evaluada 10,000 veces. Además, puede darse el caso de que la función de evaluación no

tenga una regla de correspondencia explícita, esto es, una expresión algebraica, y puede ocurrir incluso que la función cambie de generación en generación. [3]

2.1.3 Función de Adaptación

Los algoritmos genéticos son mecanismos de búsqueda que operan sobre un conjunto de códigos posiblemente muy grande, pero finito. Del dominio de búsqueda se toman iteradamente conjuntos de muestras y cada elemento de una muestra es calificado dependiendo de qué tan bien cumpla con los requisitos de aquello que es buscado. Luego son seleccionados los elementos, a los que se les llamara individuos, que cumplan mejor con dichos requisitos. Éstos se multiplican en las siguientes muestras y son sometidos a ciertos operadores que emulan a los que funcionan en la naturaleza. Uno de los elementos esenciales involucrados en este proceso es la calificación asignada a cada individuo, a la que es llamada grado de adaptación. Esta calificación es un número mayor o igual a cero, y en conjunto con la selección se convierte en una medida de la posibilidad de reproducción para cada individuo. Existe una función de adaptación que asocia a cada individuo de la muestra (población) con un número real no negativo. Mientras más grande sea el valor asignado a un individuo dado mayor será la probabilidad de éste de ser seleccionado para formar parte de la siguiente generación de muestras. También se necesitara tener un esquema, este esquema se realizara de la siguiente manera: Se ha creado una notación para los conjuntos de cadenas binarias (cromosomas) mediante cadenas compuestas de tres símbolos, a saber: {0,1,*}. Para construir la cadena que denota este conjunto, basta colocar en las posiciones donde las cadenas coinciden el valor explícito que tienen, y en las posiciones donde los valores no coinciden se coloca un *. De esta manera por ejemplo, 1*0* denota el conjunto {1000, 1001, 1100,1101}. A las cadenas compuestas de 1, 0 y * se les denomina esquemas.[4]

[3] http://www.ica.ele.puc-rio.br/pesquisa/download/paper.pdf
[4] http://www.ica.ele.puc-rio.br/pesquisa/download/paper.pdf

2.1.4 Selección.

Una vez calificados todos los individuos de una generación, el algoritmo debe, al igual que lo hacen la naturaleza y el hombre, seleccionar a los individuos más calificados, mejor adaptados al medio, para que tengan mayor oportunidad de reproducción. De esta forma se incrementa la probabilidad de tener individuos "buenos" (con una buena calificación) en el futuro. Si de una determinada generación de individuos se seleccionaran sólo aquellos con una calificación mayor o igual que el número c para pasarlos a la siguiente generación, es claro que en ésta la calificación promedio superará c y por tanto al promedio de la generación anterior. La selección ocasiona que haya más individuos buenos, explota el conocimiento que se ha obtenido hasta el momento, procurando elegir lo mejor que se haya encontrado, elevando así el nivel de adaptación de toda la población. En principio podría parecer que es conveniente tener una estrategia de selección estricta para que mejore rápidamente la población y converja el algoritmo, es decir, que la población se acumule alrededor de un genotipo óptimo. Esto no es cierto. Lo que ocurrirá es que la población se acumulará rápidamente alrededor de algún individuo que sea bueno, comparativamente con el resto de los individuos considerados a lo largo de la ejecución del algoritmo, pero este individuo puede no ser el mejor posible. A esto se le suele llamar convergencia prematura. No se puede asegurar pero sí procurar que lo anterior no ocurra. Además de la explotación es necesario que exista exploración. El AG debe, no sólo seleccionar de entre lo mejor que ha encontrado, sino procurar encontrar mejores individuos. Los que aseguran que en todo momento exista cierto grado de variedad en la población, procurando con ello que no se "vicie". En la estrategia de selección normalmente se incluye un elemento extra que sirve de "ancla". Si sólo se hace selección forzando que sea más probable elegir al mejor individuo de la población pero sin asegurarlo, es posible que este individuo se pierda y no forme parte de la siguiente generación. Para evitar esto se fuerza la selección de los

mejores n individuos de la generación para pasar intactos a la siguiente. A esta estrategia se le denomina elitismo y puede ser generalizada especificando que permanezcan en la población los n mejores individuos de las pasadas k generaciones.

2.1.5 Cruce

Durante la meiosis ocurre el proceso de producción de gametos. El código genético de los padres de un individuo se mezcla para producir gametos cuyo contenido genético es híbrido, es decir, una mezcla. De esta manera es posible que un individuo herede a sus descendientes las características mezcladas de sus propios padres, por ejemplo: el color de ojos del padre y el de cabello de la madre o, para aprovechar el ejemplo mencionado en la sección 5.3, es posible que una gacela herede la velocidad de su abuelo paterno y la fuerza de su abuela paterna, la salud de su abuelo materno y la agudeza visual de su abuela materna. Si estas características le confirieron a sus ancestros una alta aptitud de sobre vivencia, entonces este individuo será, con alta probabilidad, un individuo exitoso en su manada. La cruza de los códigos genéticos de individuos exitosos favorece la aparición de nuevos individuos que hereden de sus ancestros características deseables. En el contexto de los algoritmos genéticos reproducirse significa que, dados dos individuos seleccionados en función de su grado de adaptación, éstos pasen a formar parte de la siguiente generación o, al menos, mezclen sus códigos genéticos para generar hijos que posean un código híbrido. Es decir, los códigos genéticos de los individuos se cruzan. Existen muchos mecanismos de cruzamiento, pero todos tienen por objeto que el código de un individuo *A* y el de uno *B*, seleccionados con anterioridad, se mezclen, es decir, se fragmenten y recombinen para formar nuevos individuos con la esperanza de que éstos hereden

de sus padres las características deseables. El mecanismo de cruzamiento más común es el llamado cruzamiento de un punto. [5]

2.1.6 Mutación.

Casualmente algunos elementos del código de ciertos individuos de un algoritmo genético se alteran a propósito. Éstos se seleccionan aleatoriamente en lo que constituye el símil de una mutación. El objetivo es generar nuevos individuos, que exploren regiones del dominio del problema que probablemente no se han visitado aún. Aleatoriamente se buscan nuevas soluciones posibles que quizá superen las encontradas hasta el momento. Esta es una de las características que hacen aplicables los algoritmos genéticos a gran variedad de problemas: no reconocer conocimiento previo acerca del problema a resolver ni de su dominio, no sólo en la mutación sino en el proceso total. De hecho, el problema a resolver sólo determina la función de evaluación y la manera de codificar las soluciones posibles. El resto de los subprocesos que constituyen el algoritmo son independientes y universalmente aplicables.[6]

[5] http://www.ica.ele.puc-rio.br/pesquisa/download/paper.pdf
[6] http://www.ica.ele.puc-rio.br/pesquisa/download/paper.pdf

3 El ALGORITMO GENETICO SIMPLE (AGS)

El Algoritmo Genético Simple (AGS) propone una manera particular de seleccionar individuos y de cruzarlos. En la actualidad hay muchos métodos para trabajar con los algoritmos genéticos pero el de Holland se constituye en la base para muchos desarrollos teóricos y prácticos.

El AGS propone una codificación del dominio del problema en números binarios, este algoritmo genético tiene 13 pasos en particular. Los pasos que realiza un Algoritmo Genético Simple son los siguientes:

1. Decidir cómo codificar el dominio del problema.
2. Generar un conjunto aleatorio (población inicial) de N posibles soluciones codificadas al problema. A ésta se le llamará la población actual.
3. Calificar cada posible solución (individuo) de la población actual.
4. Seleccionar dos individuos de la población actual con una probabilidad proporcional a su calificación.
5. Lanzar una moneda al aire (con probabilidad PC cae cara).
6. Si cayó cara mezclar los códigos de los dos individuos seleccionados para formar dos híbridos, a los que llamaremos nuevos individuos.
7. Si cayó cruz llamamos a los individuos seleccionados nuevos individuos.
8. Por cada BIT de cada nuevo individuo lanzar otra moneda al aire (con probabilidad PM cae cara).
9. Si cae cara cambiar el BIT en turno por su complemento.
10. Si cae cruz el BIT permanece inalterado.
11. Incluir a los dos nuevos individuos en una nueva población.

12. Si la nueva población tiene ya N individuos, llamarla población actual y regresar al paso 3, a menos que se cumpla alguna condición de terminación.
13. Si no, regresar al paso 4.

En el algoritmo se utiliza el término “lanzar una moneda al aire” para hablar del experimento de Bernoulli (es aquel en el que pueden ocurrir exclusivamente dos eventos posibles, uno con probabilidad p y otro con probabilidad 1-p). Es decir, el lanzamiento de una moneda “extraña” en la que no necesariamente ambas caras son equiprobables. La condición de terminar el algoritmo, a la que se hace referencia en el paso 12, puede definirse de muchas maneras. Se puede fijar un número máximo de generaciones que se pretende ejecutar el algoritmo, o se puede hacer alto cuando la mayoría de la población, por ejemplo un 85%, tenga una calificación que esté dentro de 0.6 desviaciones estándar de la media. En fin, opciones hay muchas. Generalmente depende del problema o de las preferencias personales la decisión acerca de cuándo es conveniente detenerse. En el paso 4 se menciona que hay que seleccionar dos individuos con probabilidad proporcional a su calificación.[7]

[7] http://www.gsi.dit.upm.es/~gfer/ssii/aprendizaje/RN-AG.pdf

4 ALGORITMOS GENETICOS PARALELOS

La idea principal de la mayoría de los Programas Distribuidos es dividir una tarea programada. Esta división se hace con el fin de solucionar el problema planteado rápidamente y para utilizar simultáneamente múltiples procesadores.

Esta idea de división de tareas programadas se puede aplicar fácilmente a un Algoritmo Genético Simple (AGS) de muchas maneras distintas, con el fin de utilizar al máximo los recursos computacionales que existan en ese momento y así poder encontrar una solución rápida y optima al problema que se este planteando.

Los Algoritmos Genéticos Paralelos nacen de esta idea de utilizar simultáneamente múltiples procesadores para poder utilizar al máximo los recursos computacionales y encontrar la mejor solución a los problemas planteados.

Los Algoritmos Genéticos Paralelos trabajan de la misma forma que un Algoritmo Genético Simple (AGS), es decir, estos también trabajan codificando el dominio de la población (donde se pueden encontrar las mejores soluciones al problema planteado), tiene una función de evaluación la cual tiene una grado de adaptación con el cual se obtiene los mejores individuos y trabajan con los operadores de Selección, Cruce y Mutación con el fin de encontrar las mejores soluciones a los problemas propuestos.[8]

[8] http://www.ica.ele.puc-rio.br/pesquisa/download/paper.pdf

4.1 DIFERENCIA ENTRE EL ALGORITMO GENÉTICO SIMPLE Y EL ALGORITMO GENÉTICO PARALELO.

Como se ha dicho anteriormente el Algoritmo Genético Simple y el Algoritmo Genético Paralelo funcionan de la misma manera, la diferencia entre ellos radica en el número de procesadores que utiliza cada uno para poder resolver problemas del mundo real.

El Algoritmo Genético Simple (AGS) trabaja con un solo procesador en el cual se realizan todas las tareas del algoritmo como es la función de evaluación y los operadores de Selección, Cruce y Mutación.

El Algoritmo Genético Paralelo trabaja simultáneamente con múltiples procesadores en los cuales se reparte las tareas del algoritmo. Por ejemplo, en un procesador quedara funcionando el operador de selección con el fin de encontrar los mejores individuos que se acoplen al problema y en los otros procesadores quedaran los operadores de cruce y mutación que son los encargados de generar los nuevos individuos de la población ya sea cruzando los genes de los padres de los individuos o explorando regiones del dominio del problema que hasta el momento no se había conocido.

Los Algoritmos Genéticos Paralelos trabajan con una población fija de individuos o con múltiples poblaciones repartidas en varios procesadores que estén trabajando en ese momento.

La idea de trabajar con los Algoritmos Genéticos Paralelos es utilizar al máximo los recursos computacionales que existan en ese momento con el fin de encontrar una solución optima al problema que se este planteando.

4.2 DIVISIÓN DE LOS ALGORITMOS GENÉTICOS PARALELOS

Los Algoritmos Genéticos Paralelos se dividen según el numero de poblaciones que ellos utilizan. Estos algoritmos trabajan con una sola población y también trabajan con múltiples poblaciones. Los Algoritmos Genéticos Paralelos que trabajan con una sola población se conocen con el nombre de cómo Algoritmos Genéticos globales o de una sola población pues estos trabajan con una arquitectura Master esclavo y los Algoritmos que trabajan con múltiples poblaciones se conocen con el nombre de Algoritmos Genéticos Paralelos de Múltiples poblaciones.

4.2.1 Algoritmo Genético Paralelo global o de una sola población

Este tipo de Algoritmo trabaja con una sola población, se conoce también con el nombre de Algoritmo Genético Paralelo Global y la función de evaluación y el uso de los operadores genéticos se realiza en forma distribuida y en paralelo.[9]

Como en un Algoritmo Genético Simple, cada individuo de la población puede competir y cruzarse con cualquier otro individuo. El Algoritmo Genético Paralelo Global se pone en ejecución generalmente con una arquitectura Master-Esclavo, donde el Master almacena la población y los esclavos ponen en ejecución la evaluación de la población con el fin de obtener los mejores individuos.

El proceso mas común que se trabaja en el Algoritmo Genético Paralelo Global o de una sola población es la Evaluación de la población, por que con esta se conoce la función de adaptación que tiene un individuo para acoplarse a la solución del problema ya que esta es independiente del resto de la población, y

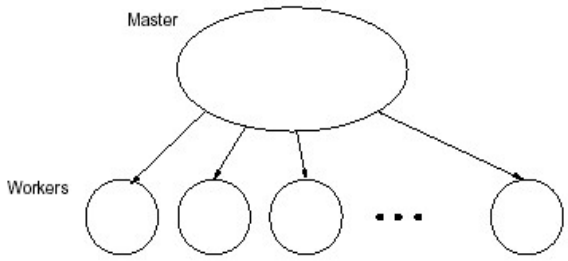
[9] http://www.fing.edu.uy/~sergion/gp/documentos/propios/EDCAGP.pdf

allí no hay ninguna necesidad de comunicación entre el Master y el Esclavo durante esta fase.

La evaluación de los individuos es hecha en paralelo asignando una fracción de la población a cada uno de los procesadores disponibles. La comunicación ocurre solamente mientras que cada esclavo recibe su subconjunto de individuos para evaluar y cuando los esclavos devuelven la calificación de la función de evaluación.

Si el Algoritmo Genético se detiene y espera para recibir la calificación de la función de adaptación, se detendrá toda la población antes de proceder en la generación siguiente, entonces se dice que el Algoritmo Genético Paralelo es sincrónico. Un Algoritmo Genético Master-Esclavo sincrónico tiene exactamente las mismas características que un Algoritmo Genético Simple, con la velocidad siendo la única diferencia. Sin embargo, es también posible establecer un Algoritmo Genético Master-Esclavo en ejecución que sea asincrónico donde el Algoritmo Genético no se detendrá para esperar ninguno de los procesadores (con esto no es lenta la comunicación), este algoritmo no trabaja igual al Algoritmo Genético Simple.

Figura 1. Un diagrama esquemático de un Algoritmo Genético Paralelo Master-Esclavo. El Master almacena una población, ejecuta operaciones del Algoritmo Genético, y distribuye unos individuos a los esclavos. Los esclavos evalúan solamente la aptitud de los individuos.



Fuente: http://www.fing.edu.uy/~sergion/gp/documentos/propios/EDCAGP.pdf

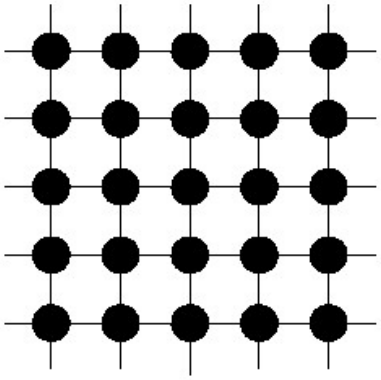
4.2.2 Algoritmos Genéticos de Múltiples poblaciones o Distribuido

El Algoritmo Genético Paralelo de Múltiple-población es un algoritmo más sofisticado, pues consisten en varias sub. poblaciones que algunas veces intercambian individuos. Este intercambio de individuos se llama migración. El Algoritmo Genético Paralelo de Múltiple-población es muy popular, pero también es el tipo de Algoritmo Genético Paralelo que es el más difícil de entender, porque los efectos de la migración no se entienden completamente. El Algoritmo Genético paralelo de Múltiple-población introduce cambios fundamentales en la operación del Algoritmo Genético y tiene un comportamiento distinto que el Algoritmo Genético Simple.

La migración de individuos de una población a otra se controla por varios parámetros, estos parámetros son los siguientes:
La topología que define las conexiones entre las sub. Poblaciones
Una tarifa de la migración que controla cuántos individuos emigran
Un intervalo de migración que afecta la frecuencia de migraciones.

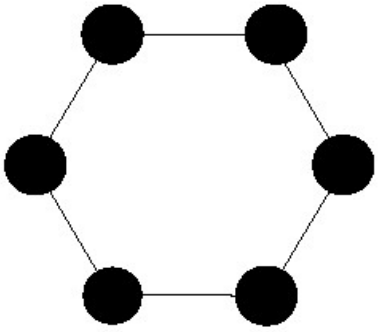
El Algoritmo Genético Paralelo Múltiple-población se conoce con diversos nombres. Se conocen a veces como Algoritmo Genético Distribuido, porque se ponen en ejecución generalmente en las computadoras de memoria distribuida.

Figura 2. Un diagrama esquemático de un Algoritmo Genético Paralelo. Esta clase del Algoritmo Genético paralelo tiene una población espacial distribuida, y puede ser puesta en ejecución muy eficientemente en las computadoras paralelas.



Fuente: http://www.fing.edu.uy/~sergion/gp/documentos/propios/EDCAGP.pdf

Figura 3. Un diagrama esquemático de un Algoritmo Genético Paralelo de la múltiple-población. Cada proceso es un Algoritmo Genético Simple, y hay comunicación entre las poblaciones.



Fuente: http://www.fing.edu.uy/~sergion/gp/documentos/propios/EDCAGP.pdf

4.2.3 Diferencias entre El Algoritmo Genético Global o de una solo población y el Algoritmo Genético Paralelo Distribuido o de múltiple población.

La diferencia entre estos dos tipos de Algoritmos Genéticos radica en la forma en que ellos trabajan ya que el método de una sola población no afecta el comportamiento del algoritmo, pero el método de múltiple población si afecta la manera que el algoritmo trabaja.

Por ejemplo, en el Algoritmo Genético Paralelo de una solo población la selección se realiza a toda la población, en el Algoritmo Genético Paralelo de múltiple-población la selección solamente se realiza a un subconjunto de individuos. También, en el algoritmo de una sola población cualquier individuo se puede cruzar con otro (es decir, hay cruce al azar), pero en el algoritmo de múltiple-población el cruce se restringe a un subconjunto de individuos.

Hay un Algoritmo Genético Paralelo que reúne estos dos algoritmos. Este se llama Algoritmo Genético Paralelo Jerárquico, Pues este combina las ventajas de los métodos anteriores y propone un funcionamiento mejor que los anteriores.[10]

[10] http://www.fing.edu.uy/~sergion/gp/documentos/propios/EDCAGP.pdf

5. PROGRAMACION DISTRIBUIDA

La programación distribuida generalmente significa la conexión de un cliente a un servidor donde se encuentra la información que es necesaria para poder trabajar con cierta información que allí se encuentra. El problema con esta arquitectura es que si se pierde la conexión con el servidor, los clientes no pueden utilizar la información que se encuentra en este.

Para evitar esta pérdida de tiempo, se crearon los diferentes modelos de red. Un ejemplo es el modelo de servidor maestro y esclavo donde si el maestro falla, el esclavo toma el relevo.

El problema con los distintos modelos de red es que todos requieren alguna forma de intervención manual y se unieron con un sistema operativo o un lenguaje. Y aunque estas aproximaciones consiguieron reducir el tiempo de parada, no cumplen con los sistemas de distribución heterogénea que consiste en una mezcla de protocolos de red y máquinas.

La plataforma Java combinada con otros avances como Common Object Request Broker Architecture (CORBA), servidores multi-fila, y redes sin cables han llevado un paso mas allá la realización de la computación totalmente distribuida, de la tradicional aproximación cliente y servidor.

Algunos elementos de la Programación Distribuida son el servicio de búsqueda, RMI, JDBC entre otos.

5.1 SERVICIO DE BÚSQUEDA

Los servicios de búsqueda permiten las comunicaciones a través de la red. Un programa cliente puede usar un protocolo de búsqueda para obtener información sobre programas remotos o máquinas que usen esa información para establecer una comunicación. Algunos protocolos de búsqueda proporcionan servicios de directorio. Este servicios como el Lightweight Directory Access Protocol (LDAP) y el NIS+ de Sun proporcionan otra información y servicios más allá de los disponibles con el servicio de nombres. Por ejemplo, NIS+ asocia un atributo workgroup con una cuenta de usuario. Este atributo puede usarse para restringir el acceso a una máquina, por lo que sólo los usuarios especificados en el workgroup tienen acceso.

5.2 RMI (Remote Method Invocation)

La Invocación Remota de Métodos (RMI) permite las comunicaciones entre cliente y servidor a través de la red entre programas escritos en Java. El RMI nos permite accede a un servidor de objetos remoto desde un programa cliente haciendo sencillas llamadas a métodos del servidor de objetos. Mientras que otras arquitecturas distribuidas para acceder a servidores de objetos remotos como “Distributed Component Object Model” (DCOM) y “Common Object Request Broker Architecture” (CORBA) devuelven referencias al objeto remoto, el RMI no sólo devuelve referencias, si no que proporciona beneficios adicionales. El RMI maneja referencias a objetos remotos (llamadas por referencia) y también devuelve una copia del objeto (llamada por valor). Si el programa cliente no tiene acceso local a la clase para la que se ejemplarizó un objeto remoto, los servicios RMI pueden descargar el fichero class.

5.3 JDBC

El JDBC se utiliza para conectarse a las bases de datos utilizando el lenguaje de programación java. Por defecto, el acceso a bases de datos JDBC implica abrir una conexión con la base de datos, ejecutar comandos SQL en un sentencia, procesar los datos devueltos y cerrar la conexión con la base de datos. En conjunto, la aproximación por defecto funciona bien para bajos volúmenes de acceso a la base de datos.[11]

[11] http://programacion.com/java/tutorial/jdcbook/5/#jdcbookcalculodistribuido

6. DESARROLLO DEL PROYECTO

6.1 ANÁLISIS Y DISEÑO

6.1.1 Declaración del problema

La Universidad Autónoma de Bucaramanga requiere un modulo de software que facilite la automatización de la elaboración de los horarios de las asignaturas de los diferentes programas académicos de la universidad.

La programación de los horarios de las asignaturas de cada una de las facultades que conforman la Universidad es un problema multivariable que involucra optimización de recursos. Entre las variables determinantes se encuentran la cantidad de docentes, su disponibilidad horaria, las asignaturas que pueden impartir; la cantidad, capacidad y disponibilidad horaria de los salones; entre otras. Además existen ciertas restricciones que complican aún más esta tarea como por ejemplo: las franjas horarias según los niveles, evitar los cruces de asignaturas que pertenecen a un mismo nivel, evitar los cruces de los horarios de los docentes, etc.

Esto en la actualidad se hace con ayuda de herramientas ofimáticas como Word y Excel, causando inconsistencias al momento de integrar toda la información.

El desarrollo de un algoritmo tradicional para la solución de este tipo de problema resultaría bastante complejo, por lo que se utilizarán técnicas de algoritmos genéticos que simulan el proceso de evolución natural.

Para poder elaborar los horarios de los docentes el Coordinador de Facultad o el Jefe de Departamento debe suministrar la siguiente información:

Disponibilidad horaria de los docentes a su cargo

Si el docente es de tiempo completo su disponibilidad corresponde a la totalidad de horas estipuladas en su jornada laboral y aquellas horas por fuera de su horario normal siempre y cuando se cuente con la aprobación del docente. Si el docente es de hora cátedra su disponibilidad no debe ser inferior al total de horas de la carga asignada para el semestre a programar.

Carga académica de cada docente

Indica cuantas secciones de cada curso corresponden a cada docente y se elabora con base en la demanda de cursos proyectada para el semestre a programar. La demanda puede establecerse mediante consultas al sistema BANNER y datos estadísticos acerca del ingreso de estudiantes nuevos durante los últimos semestres.

Horarios Preestablecidos

Indica aquellos horarios de reunión que el sistema debe respetar ya que no pueden ser modificados por diversas circunstancias como disponibilidad del docente, recursos, etc.

Adicionalmente se debe extraer del sistema BANNER la siguiente información:

Datos de los docentes

Información básica como el ID, Apellidos, Nombres, Dedicación, Dependencia.

Datos de los salones

Código, Campus, Edificio, Salón, Capacidad, Tipo de Salón (dependiendo de los recursos con que cuenta).

Información referente a los planes de estudios de los diferentes programas.

Todos los salones sin excepción tendrán una disponibilidad de 6:00 AM a 10:00 PM de lunes a domingo.

6.2 ANÁLISIS DE REQUERIMIENTOS DEL SISTEMA DE PROGRAMACIÓN DE HORARIOS

El análisis de requerimientos describe toda la información recolectada para que el sistema trabaje correctamente, en este análisis se describen los requerimientos funcionales y los requerimientos no funcionales del sistema de programación de horarios.

6.2.1 Requerimientos Funcionales del Sistema de Horarios

Los requerimientos funcionales describen las interacciones entre el sistema y su ambiente en forma independiente a su implementación.

Elaborar un software que ofrezca las siguientes facilidades:

Desarrollar un software que permita la elaboración de los horarios de cada uno de los docentes que conforman la Universidad Autónoma de Bucaramanga utilizando técnicas de algoritmos genéticos.

El sistema debe tener en cuenta la información de los docentes, esta información es la siguiente: Nombres del docente, Apellidos del docente, disponibilidad horaria del docente, cursos a dictar por cada docente, dedicación del docente, número de horas del curso, requisitos del curso a dictar por cada docente, correquisitos del curso, semestre al que pertenece el curso, programa al que pertenece cada curso.

El sistema también debe respetar la información de los docentes que no se puede cambiar, como las franjas horarias fijas que los docentes tienen para dictar sus cursos.

El sistema debe tener en cuanta la información de los salones, esta información es la siguiente: La ubicación de los salones es decir en campus y edificio de cada campus de la Universidad se encuentra ubicado, número de salones que hay en la universidad, de estos salones cuantos hay disponibles para dictar los cursos de cada programa, tipo de salón, capacidad de los salones.

El sistema debe respetar las reuniones de las secciones de los cursos, es decir las horas exactas de cada reunión de una sección de un curso a ser dictada, como por ejemplo una hora o dos horas exactas.

El sistema de matriculas BANNER suministrará la información de los salones y la de los docentes que sea necesario, esta información es la mencionada anteriormente.

El sistema también debe permitir la captura de la información desde otra base de datos, esta es la base de datos del sistema, esta información es la franja horaria y disponibilidad horaria suministrada por cada docente para dictar sus cursos.

El sistema utiliza técnicas de Algoritmos Genéticos para poder realizar la tarea de generar los horarios de los docentes.

El sistema empieza con la codificación del dominio del problema, después de esto selecciona la población inicial, ya que por medio de esta solución es mas fácil poder encontrar el mejor horario posible, después genera una función de evaluación con el fin de encontrar las mejores soluciones y así poder generar la siguiente generación con los horarios mas capacitados, después de esto se generan los operadores de cruce y mutación con el fin de encontrar individuos mas aptos que puedan así encontrar mas rápido una solución a nuestro problema de los horarios de los docentes.

6.2.2 Requerimientos no Funcionales

Los requerimientos no funcionales describen aspectos del sistema que son visibles por el usuario pero que no se relacionan directamente con el comportamiento funcional del sistema.

Homogeneidad:

Lenguaje: El sistema debe desarrollarse en JAVA.
Nivel usuario: Se refiere al nivel de seguridad para el acceso al sistema.
Sencillez. Facilidad de uso y entendimiento.

Flexibilidad

Posibilitar configuración.

Portabilidad

Se instala sobre cualquier plataforma.

6.2.3 Seudorequerimientos

Todo el software se escribirá usando el lenguaje de programación JAVA para cumplir con la política del director del proyecto.

6.2.4 Escenarios

Nombre del escenario	**Generar Horario de los Docentes**
Participantes	**Coordinador de Facultad o Jefe de Departamento**
Flujo de eventos	1. El coordinador de facultad o el jefe de departamento es el encargado de inicia el proceso de la elaboración de los horarios de los docentes. 2. El coordinador de facultad o jefe de departamento es el encargado de consultar los horarios que van hacer publicados y este es el que mira si el horario sirve o no. 3. El coordinador de facultad o jefe de departamento es el encargado de modificar los horarios que no cumplan con los requerimientos iniciales propuestos. 4. El coordinador de Facultad o el jefe de departamento es el encargado de ingresar los datos del docente a la base de datos del sistema, esta información es la disponibilidad horaria del docente para dictar sus cursos .

Nombre del escenario	**Generar Horarios de los Docentes**
Participantes	**Sistemas de Matriculas BANNER**
Flujo de eventos	1. El Sistema de matriculas BANNER es el encargado de suministra los datos del docente a la persona encargada de utilizar el sistemas para generar los horarios, esta información es la siguiente: - Nombres del docente. - Apellidos del docente. - Dedicación del docente - Cursos a dictar por cada docente. - Numero de horas de cada curso. - Requisitos de los cursos. - Correquisitos de los cursos. - Programa a la que pertenece cada curso. - Semestre a la que pertenece cada curso. - Numero de sesiones de cada curso. 2. El sistema de matriculas BANNER es el encargado de suministra la información de los salones a las personas encargadas de utilizar el sistema de Horarios, esta información es la siguiente:

	- Ubicación de los salones. - Numero de salones disponibles para dictar los cursos a programar. - Capacidad de cada salón. - Tipo de salón.
Nombre del escenario	**Generar Horarios de los Docentes**
Participantes	Base de Datos
Flujo de eventos	1. La base de datos del sistema es la encargada de suministrar la disponibilidad horaria de cada docente para dictar sus cursos.

6.3 CASOS DE USO DEL SISTEMA DE PROGRAMACIÓN DE HORARIOS.

6.3.1 Actores.

Coordinador de Facultad o Jefe de Departamento

El actor coordinador de o jefe de departamento es el encargado de iniciar el proceso de la elaboración de los horarios de los docentes, también es el encargado de consultar los horarios de los docentes y modificarlos en el caso que no cumplan con los requisitos iniciales.

Sistema de Matriculas BANNER

El actor sistema de matriculas BANNER es el encargado de suministrar la información necesaria de los docentes y de los salones a los coordinadores de facultad o a los jefes de departamento para que el software funcione correctamente.

6.3.2 Casos de Uso

Ingresar Datos

El Coordinador de facultad o el jede de departamento es el encargado de ingresar los datos del docente al sistema, el coordinador de facultad o el jefe de departamento también se encarga de ingresar los datos del docente a la base de datos del sistema, esta información es la disponibilidad horaria y franjas horarias de cada docente. El sistema de matriculas BANNER es el encargado de suministrar al coordinador de facultad o jefe de departamento los datos de los

salones que se necesiten para poder generar los horarios, con el fin que el coordinador de facultad o jefe de departamento ingrese esta información al sistema de horarios.

Buscar Datos

El coordinador de facultad o el jefe de departamento es el encargado de buscar los datos en la base de datos del Sistema, esta búsqueda se realiza cuando se necesita alguna información que se encuentra en la base de datos del sistema de Horarios.

Eliminar Datos

El Coordinador de facultad o el jefe de departamento es el encargado de eliminar los datos de la base de datos del sistema, los datos se eliminan cuando estos no son importantes para el sistema, por ejemplo cuando un docente ya no dictar clases en la universidad o cuando un curso deja de pertenecer a un programa.

Actualizar Datos

El Coordinador de facultad o el jefe de departamento es el encargado de actualizar los datos en la base de datos del sistema, esta operación se realiza cuando algún dato quedo mal agregado en la base de datos, por ejemplo el nombre de algún docente, el apellido de algún docente, el código de un requisito de un curso etc.

Generar Horarios

El Coordinador de facultad o jefe de departamento es el encargado de iniciar el proceso de generar los horarios de los docentes.

Consultar Horarios

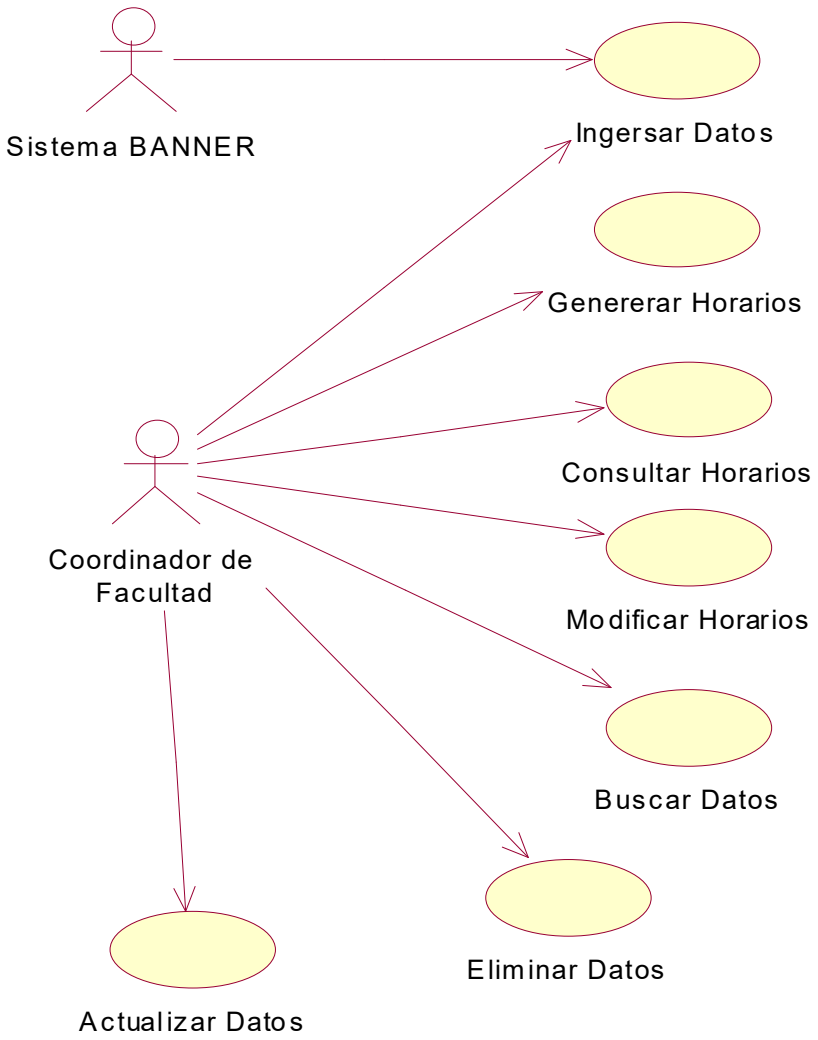
El Coordinador de facultad o jefe de departamento es el encargado de consultar los horarios que ya han sido generados con el fin de mirar si los horarios son buenos o malos.

Modificar Horarios

El Coordinador de facultad es el encargado de modificar los horarios que no cumplen con los requerimientos iniciales del sistema

6.3.3 Diagrama de Casos de Uso (Modulo de Generación de Horarios)

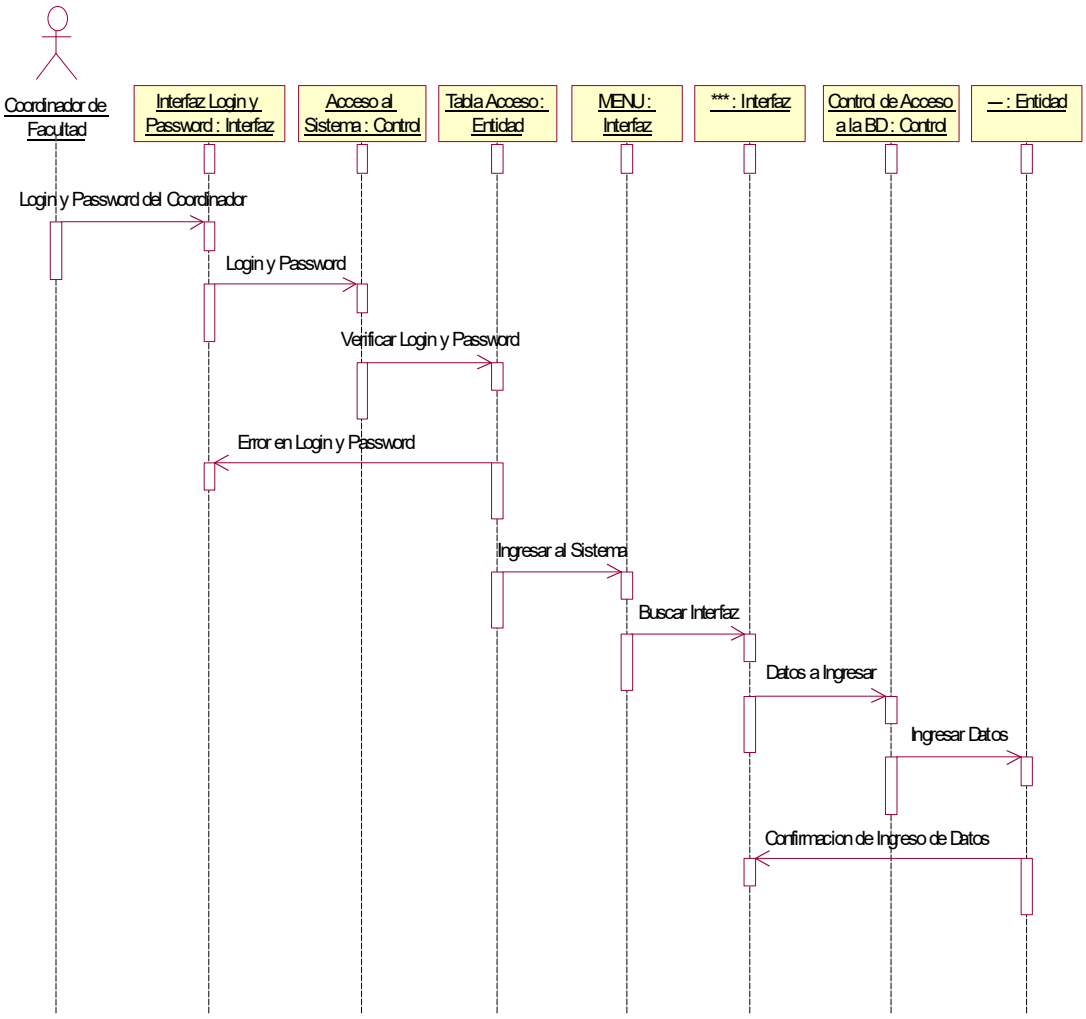
Figura 4 Diagrama de Casos de Uso Generar Horarios



6.3.4 Diagramas de Secuencia del sistema de Horarios.

6.3.4.1 Diagrama de Secuencia del caso de uso Ingresar Datos.

Figura 5 Diagrama de Secuencia Ingresar Datos



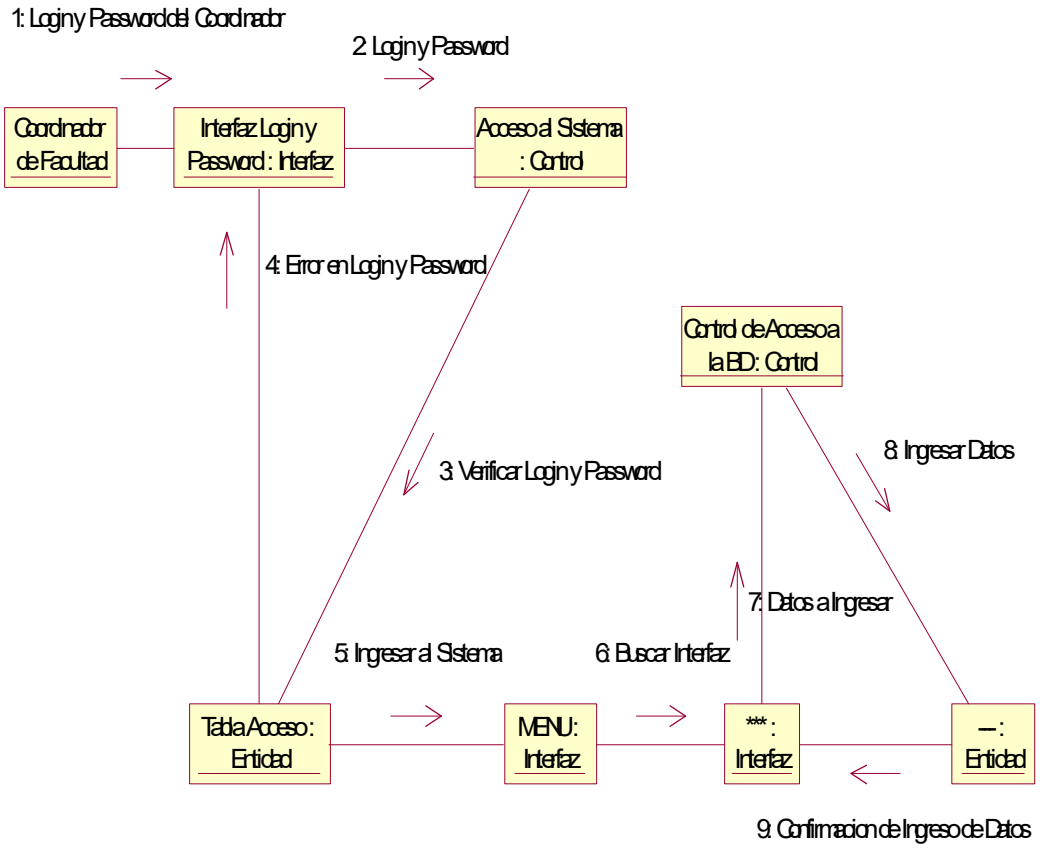
Donde:

**** Es la interfaz en la cual se ingresan los datos en el sistema de horarios.

--- Es la entidad en la cual se introducen los datos al sistema de horarios.

6.3.4.1.1 Diagrama de Colaboración del caso de uso Ingresar Datos

Figura 6 Diagrama de Colaboración Ingresar Datos



6.3.4.2 Diagrama de Secuencia del caso de uso Buscar Datos

Figura 7 Diagrama de Secuencia Buscar Datos

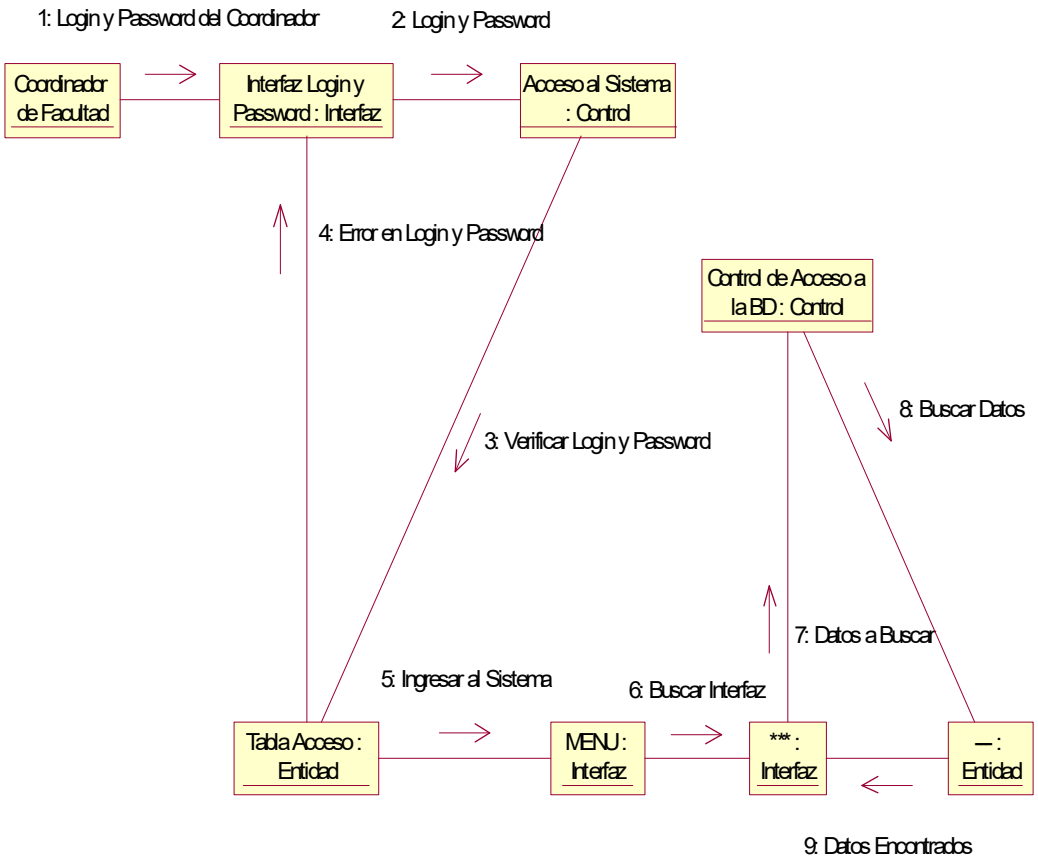
Donde:

**** Es la interfaz en la cual se van a buscar los datos en el sistema de horarios.

--- Es la entidad en la cual se encuentran los datos a buscar en el sistema de horarios.

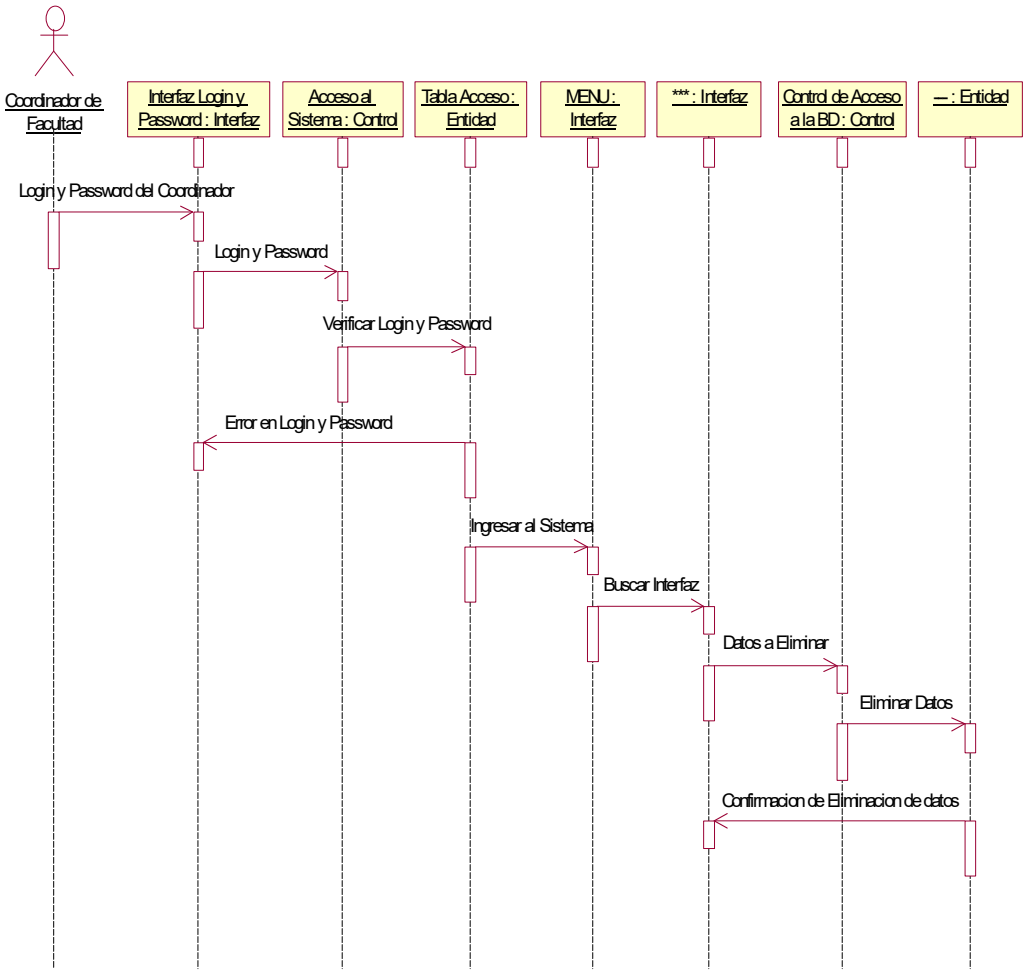
6.3.4.2.1 Diagrama de Colaboración del caso de uso Buscar Datos

Figura 8 Diagrama de Colaboración Buscar Datos



6.3.4.3 Diagrama de Secuencia del caso de uso Eliminar Datos.

Figura 9 Diagrama de Secuencia Eliminar Datos



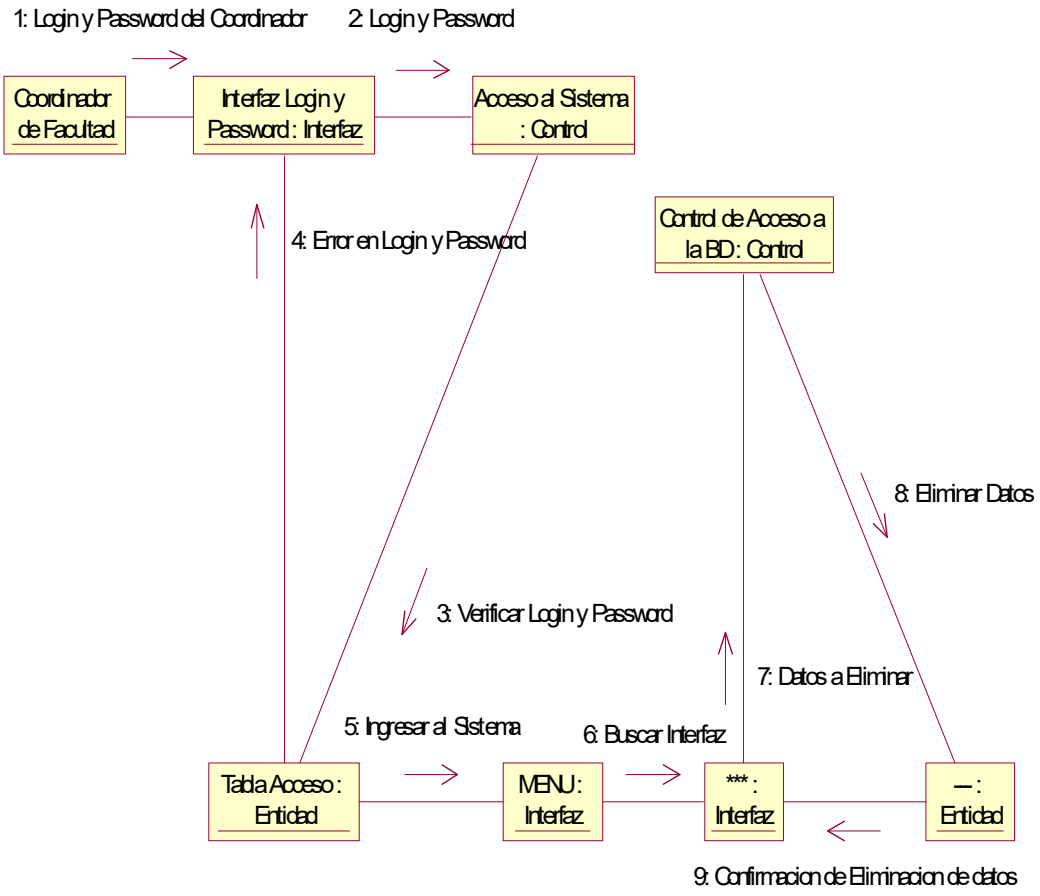
Donde:

**** Es la interfaz en la cual se van a buscar los datos para eliminarlos del sistema de horarios.

--- Es la entidad en la cual se encuentran los datos que van hacer eliminados del sistemas.

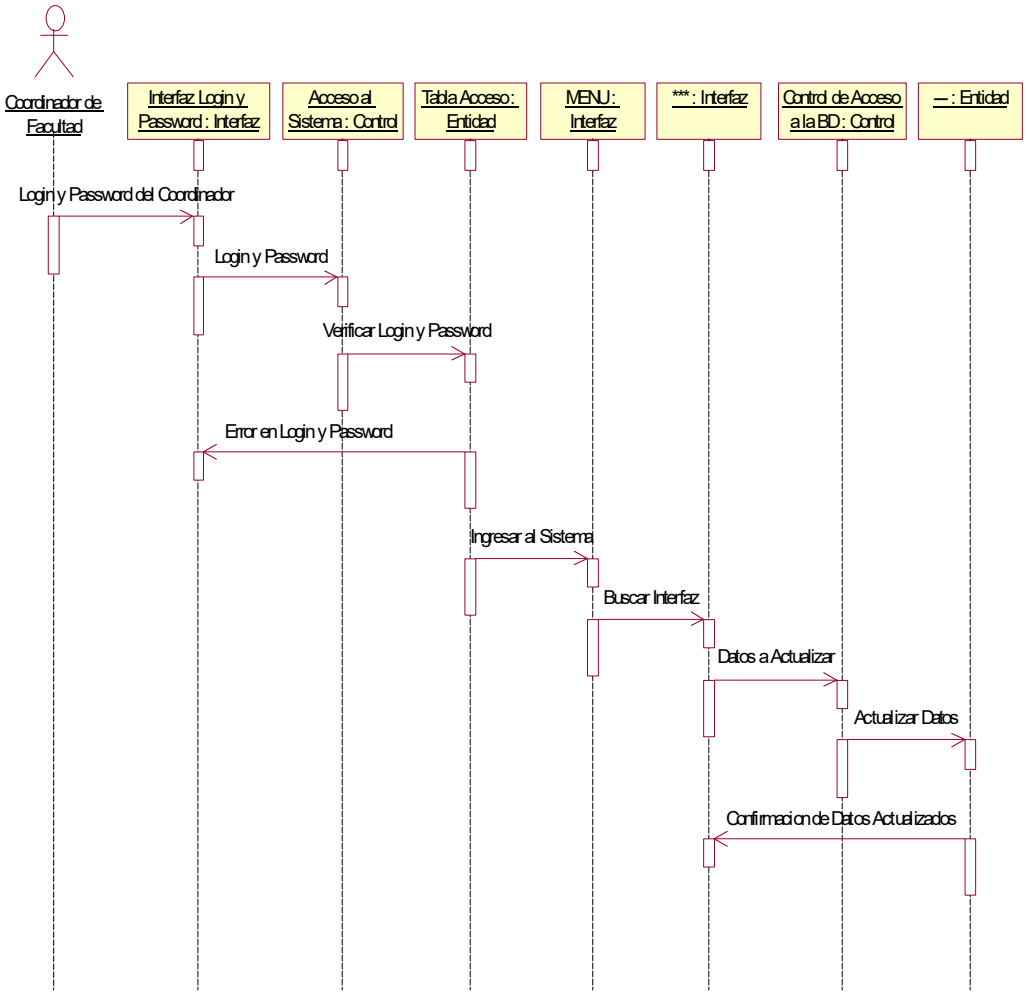
6.3.4.3.1 Diagrama de Colaboración del caso de uso Eliminar Datos.

Figura 10 Diagrama de Colaboración Eliminar Datos



6.3.4.4 Diagrama de Secuencia del caso de uso Actualizar Datos.

Figura 11 Diagrama de Secuencia Actualizar Datos



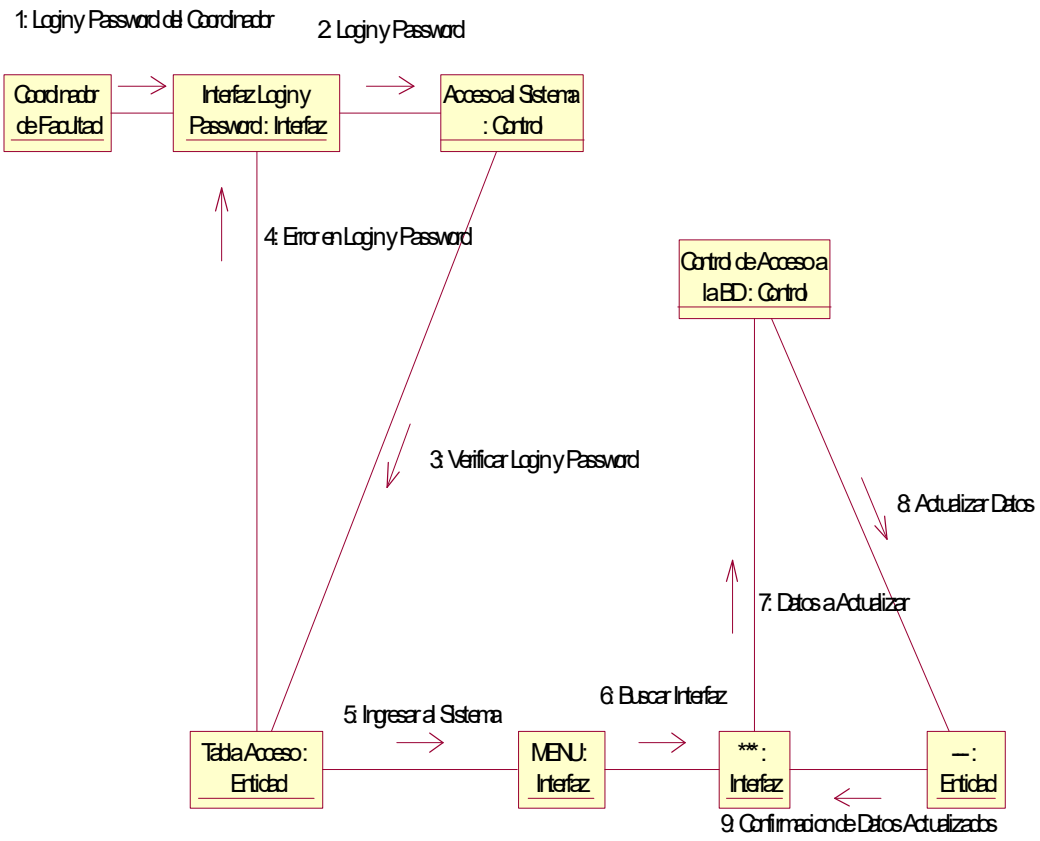
Donde:

**** Es la interfaz en la cual se van a actualizar los datos para en sistema de horarios.

--- Es la entidad en la cual se encuentran los datos que van hacer actualizados en el sistemas horarios.

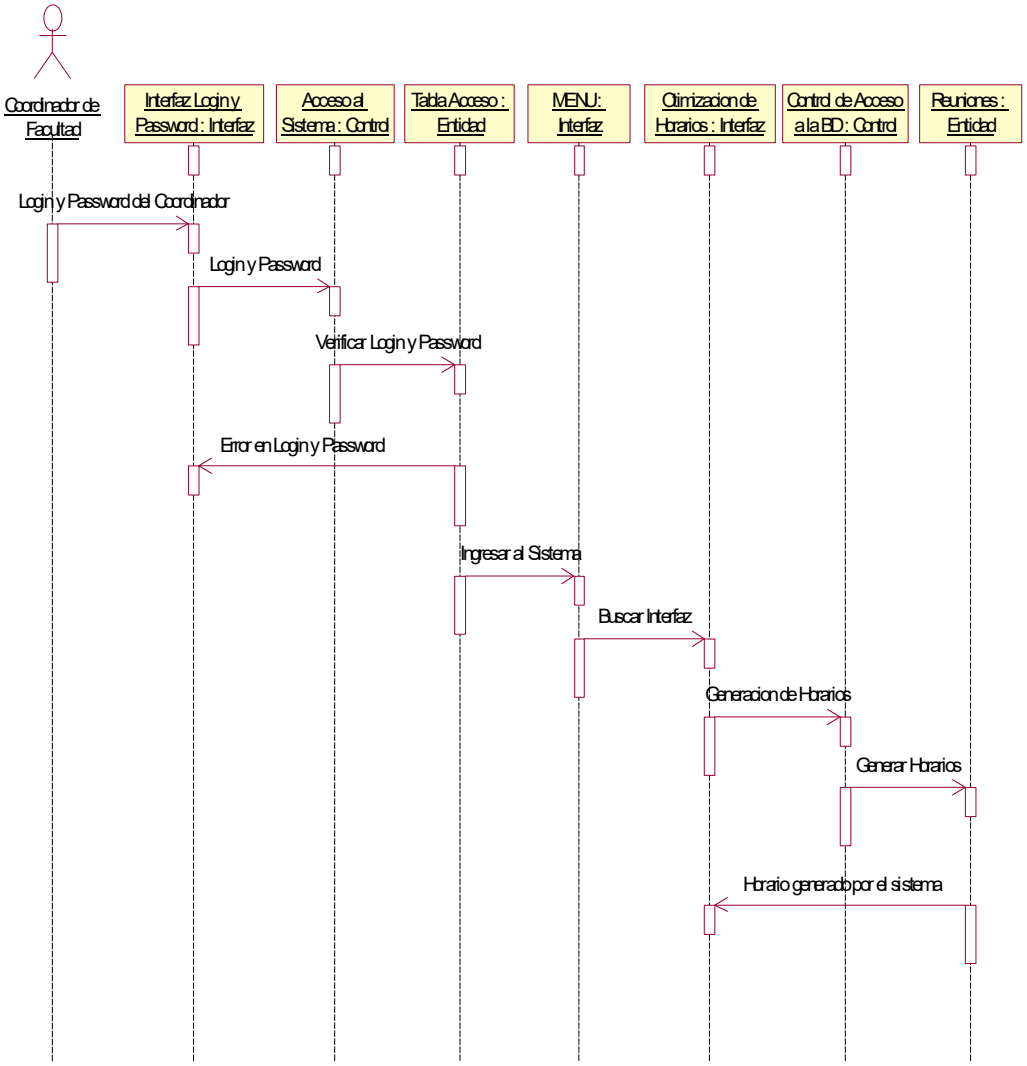
6.3.4.4.1 Diagrama de Colaboración del caso de uso Eliminar Datos.

Figura12 Diagrama de Colaboración Eliminar Datos



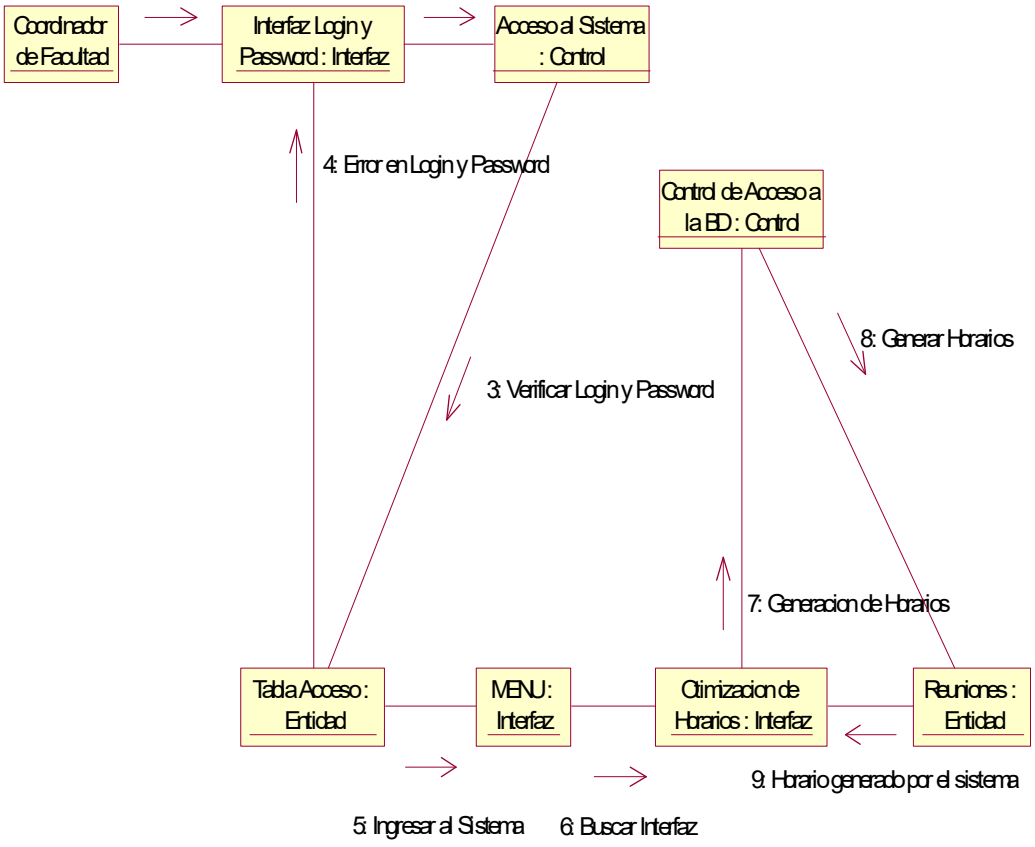
6.3.4.5 Diagrama de Secuencia del caso de uso Generar Horarios.

Figura 13 Diagrama de Secuencia Generar Horarios



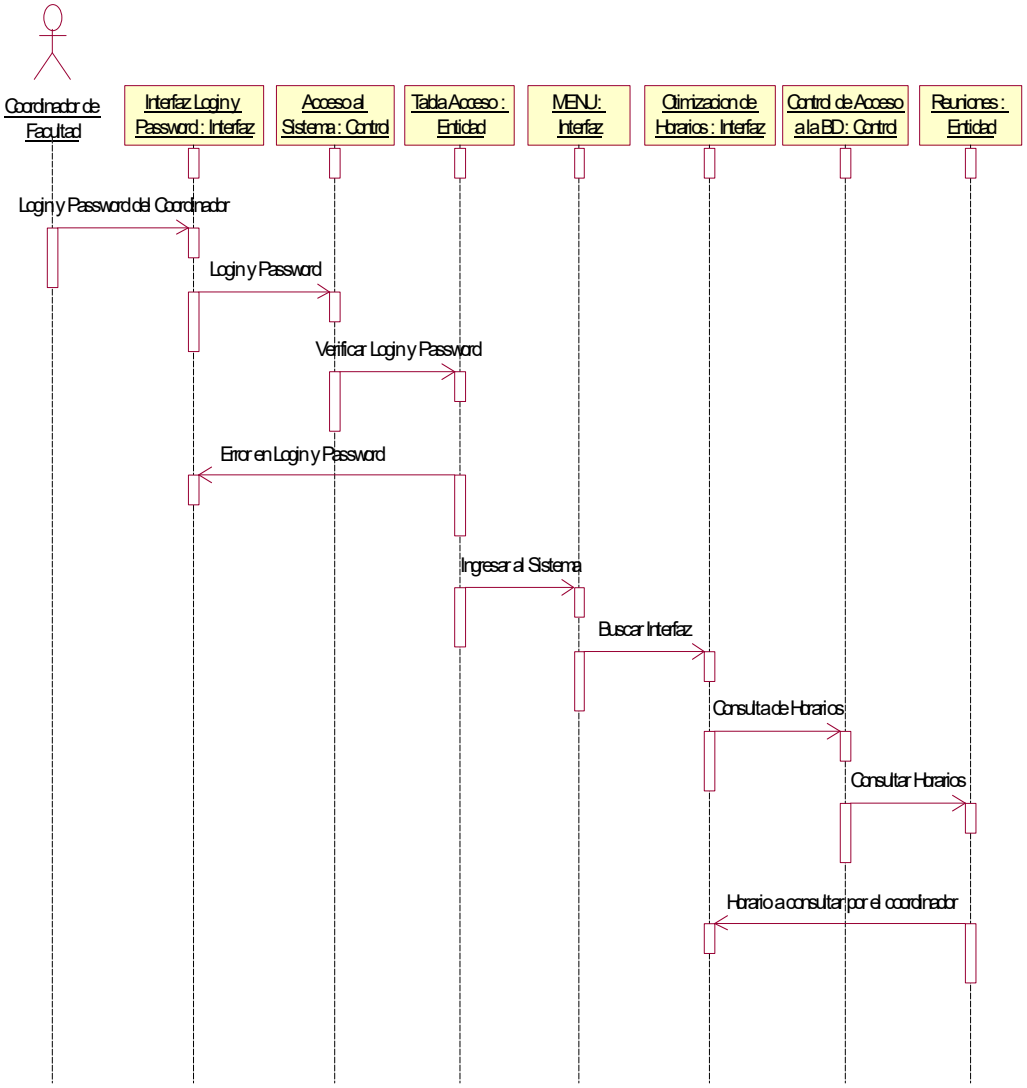
6.3.4.5.1 Diagrama de Colaboración del caso de uso Generar Horarios.

Figura 14 Diagrama de Colaboración Generar Horarios



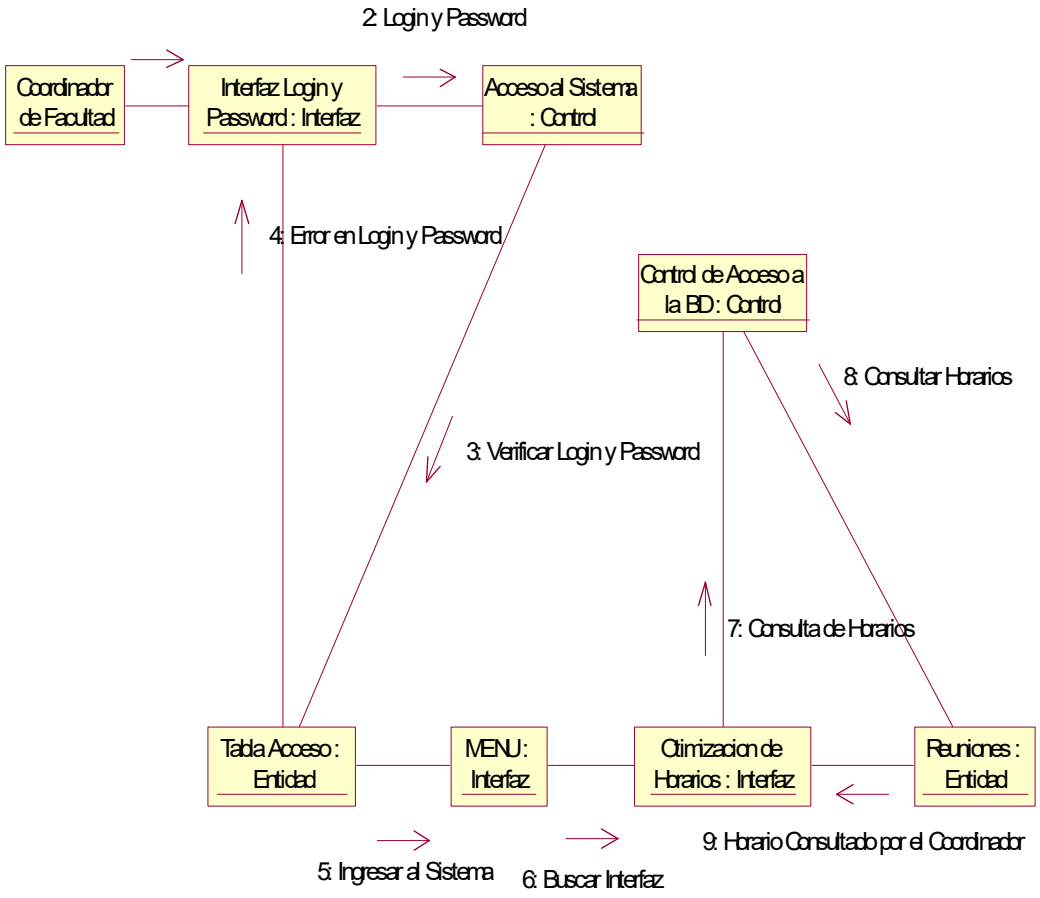
6.3.4.6 Diagrama de Secuencia del caso de uso Consultar Horarios.

Figura 15 Diagrama de Secuencia Consultar Horarios



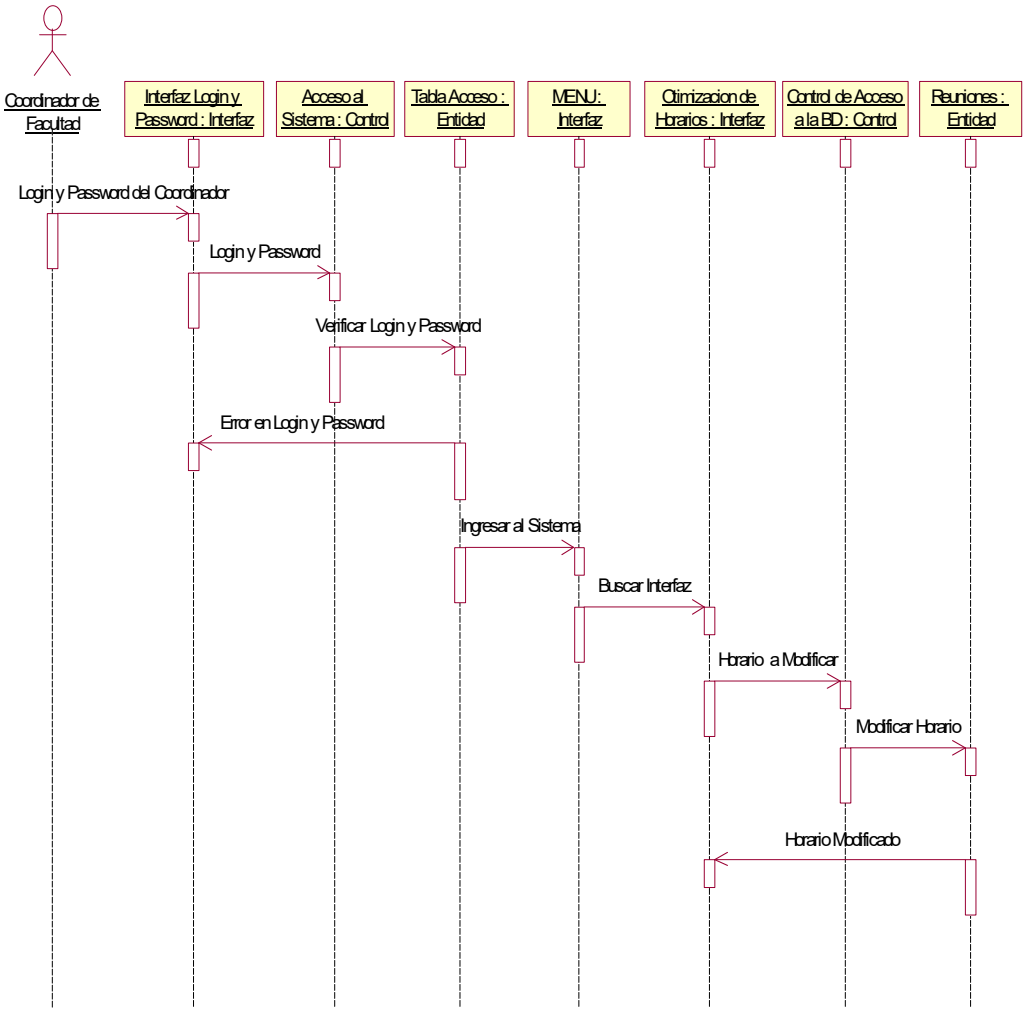
6.3.4.6.1 Diagrama de Colaboración del caso de uso Consultar Horarios.

Figura 16 Diagrama de Colaboración Consultar Horarios



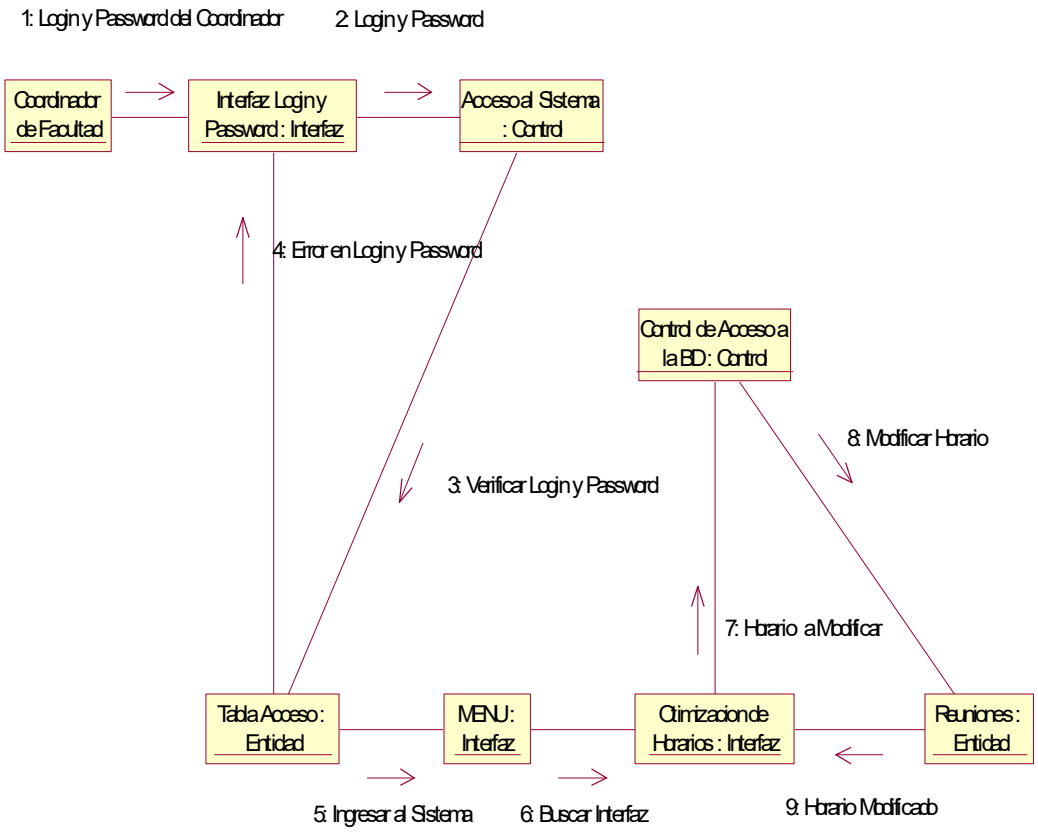
6.3.4.7 Diagrama de Secuencia del caso de uso Modificar Horarios.

Figura 17 Diagrama de Secuencia Modificar Horarios



6.3.4.7.1 Diagrama de Colaboración del caso de uso Consultar Horarios.

Figura 18 Diagrama de Colaboración Modificar Horarios



7. DICCIONARIO DE DATOS

El diccionario de datos es una descripción de todos las tablas que conforman la base de datos del sistema de programación de horarios, también describe los campo de cada tabla existente en el sistema.

La base de datos del sistema de programación de horarios tiene las siguientes tablas:

1. **Campus:** En esta tabla esta disponible toda la información de los campus que conforman la Universidad Autónoma de Bucaramanga, por ejemplo, ubicación del campus, nombre. Esta tabla contiene los siguientes atributos:

   IdCampus: Describe el identificador del campus de la universidad.

   CAMP_CODE: Describe la sigla del campus de la universidad con la cual se conoce este, este es el código que le asigna el sistema de matriculas BANNER a los campus en general. Esto se realiza con el fin que el esta tabla del sistema de horarios sea compatible con campos específicos del sistema de matriculas BANNER.

   Campus: En este campo establece el nombre del campus de la universidad.

2. **Clases:** Esta tabla indica toda la información del nivel al cual pertenece un curso, es decir en el caso del pregrado en que semestre esta ubicado el curso a programar. Esta tabla tiene los siguientes campos:

   IdClase: Se establece el identificador de la clase.

   CLAS_CODE: Describe el identificador de la clase, este es el código que le asigna el sistema de matriculas BANNER a las clase en general. Esto se realiza con el fin que el esta tabla del sistema de horarios sea compatible con campos específicos del sistema de matriculas BANNER.

   Clase: Describe el nivel donde esta ubicado el curso a programar, por ejemplo con un curso de pregrado se describe el nombre del semestre donde se encuentra este curso ubicado.

3. **Correquisitos:** Esta tabla contiene toda la información de los correquisto que necesita el curso para ser programado. El correquisito de un curso es otro curso del mismo semestre que necesita este para poder ser programado, por ejemplo, Física I necesita del laboratorio de Física I para programarse, es decir Laboratorio de Física I es correqusito de Física I. Esta tabla tiene los siguientes campos:

   IdCursoPrograma: Describe el identificador del curso por programa, con este identificador se establece cual es el curso que necesita los corequisitos para ser programado.

   IdCorrequisito: Se describe el identificador del correquisito que necesita el curso para ser programado.

4. **CorrequisitosEP:** Esta tabla contiene toda la información de los correquisitos de las Electivas Profesionales que la universidad tiene para dictar. Esta tabla tiene los siguientes atributos:

   IdElectivaProfesional: Este campo describe el identificador de la Electiva Profesional la cual se le establecerán los correquisitos.

   IdCorrequisito: Este campo estable es identificador del correquisito de l Electiva Profesional.

5. **Cursos:** En esta tabla se encuentra toda la información de los cursos a ser programados. Esta tabla tiene los siguientes campos:

   Idseccion: Se establece el identificador del curso a ser programado, esta es la llave primaria de esta tabla.

   IdLinea: Este idenificador establece la línea a la cual pertenece l curso hacer programado.

   CRSE_NUMBER: Establece la siglas del área a la cual pertenece el curso a ser programado, esta siglas las asigna el sistema de matriculas BANNER a los cursos en general. Esto se realiza con el fin que el sistema de horarios sea compatible con campos específicos del sistema de matriculas BANNER.

   Curso: Describe el nombre del curso a ser programado.

   Créditos: Describe el numero de créditos que tiene l curso a programar.

HorasTeoricas: Se establece el número de horas teóricas que tiene el curso a programar.

HorasPracticas: Se establece el número de horas prácticas que tiene el curso a programar.

CapacidadMaxima: Se indica la capacidad máxima que tiene un curso para recibir a sus estudiantes, con el fin de no exceder la capacidad de este curso.

6. **SeccionesSimultaneas:**

IdProgramaResponsable: Describe el identificador del programa que es responsable del curso a programar, con este se conoce el programa que es dueño del curso.

ClaseAprobada: Describe el identificador de la clase a la cual pertenece el curso, es decir en que semestre esta ubicado este.

7. **CursosPrograma:** Esta tabla se construye con el fin de romper la relación muchos a muchos que existía entre la tabla Programa y la tabla Cursos. Esta tabla contiene los siguientes atributos:

IdCursoPrograma: Describe el identificador del curso por programa que se va a programar.

IdPrograma: Se establece el identificador del programa, con este se conoce el programa al cual pertenece el curso que será programado.

Idseccion: Describe el identificador del curso, con este se conoce el curso que será programado.

IdClase: Se establece el identificador de la clase, con este se conoce el nivel al que pertenece el curso, por ejemplo si es un curso de pregrado, este nivel será el semestre donde esta ubicado el curso.

8. **Dedicacion:** Se establece el tipo de contrato que tiene el docente con la Universidad, es decir, si es de medio tiempo, tiempo completo u hora cátedra. Esta tabla tiene los siguientes atributos:

   IdDedicacion: Describe el identificador de la dedicación del docente.

   FSTP_CODE: Se establecen las siglas según el tipo de contrato que tenga el docente con la Universidad, es decir si es tiempo completo, medio tiempo u hora cátedra. Estas siglas son asignadas por el sistema de matriculas BANNER para todos los docentes en general. Esto se realiza con el fin que esta tabla del sistema de horarios sea compatible con campos específicos del sistema de matriculas BANNER

   Dedicación: Se establece el nombre del tipo de contrato que tiene el docente con la universidad, con este se conoce si le docente es de hora cátedra, tiempo completo o medio tiempo.

   MinimoHorasDisponibilidad: Establece el mínimo de horas disponibles que el docente tiene para dictar un curso.

9. **DemandaCancelaciones:** Esta tabla se construye con el fin de conocer la demanda de cancelaciones de los diferentes cursos de la universidad. Esta tabla contiene los siguientes campos:

TERM_CODE_KEY: En este campo se establece el identificador del curso hacer programado, este es el código que le asigna el sistema de matriculas BANNER a las cancelaciones en general. Esto se realiza con el fin que el esta tabla del sistema de horarios sea compatible con campos específicos del sistema de matriculas BANNER.

IdPrograma: Establece el identificador del programa al cual pertenece el curso cancelado.

Idseccion: Establece el identificador del curso cancelado.

Curso: Establece el nombre del curso cancelado.

Cancelaciones: Establece el numero de cancelaciones de cada curso en un semestre

10. **DiasSemana:** En esta tabla se indica la información de los días de la semana en los cuales la universidad esta libre para poder programar los cursos.

IdDiaSemana: Describe el identificador del día en el cual la universidad esta libre para programar los cursos.

Dia: Se establece el nombre del día de la semana que la universidad esta libre para programar los cursos.

Abreviatura: Describe la abreviatura del día de la semana en el cual la universidad esta libre para programar lo cursos

11. **DisponibilidadDocente:** Esta tabla indica la información de la disponibilidad horaria de lo docentes para dictar sus cursos. Este tabla tiene los siguientes campos:

    IdDisponibilidad: Describe el identificador de la disponibilidad del docente para impartir sus cursos.

    IdDocente: Describe el identificador del docente, con este se conoce cual es el docente al que pertenece la disponibilidad horaria.

    IdDiaSemana: Describe el identificador del día de la semana, con este se conoce que días de la semana el docente puede impartir sus cursos.

    HoraInicio: Se establece la hora de inicio de la disponibilidad horaria del docente.

    HoraFin: Se establece la hora final de la disponibilidad horaria del docente.

12. **Docentes:** En esta tabla se encuentra toda la información de los docentes a los cuales se les programara el horario. Esta tabla tiene los siguientes campos:

    IdDocente: Describe el identificador del docente al cual se le programara el horario.

    Nombres: Se establece los nombres del docente**.**

    Apellidos: Se establece los apellidos del docente.

IdDedicacion: Describe el identificador de la dedicación, con este se conoce el tipo de contrato que tiene el docente con la universidad ya que esta información es vital para poder programar el horario del docente. Con esto se conoce si el docente es de hora cátedra, medio tiempo o tiempo completo

13. **DocentesPrograma:** Esta tabla se construye con el fin de romper la relación muchos a muchos que existía entre la tabla Docentes y la tabla programa. Esta tabla tiene los siguientes campos:

    IdPrograma: Describe el identificador del programa, con este se conoce el programa al que hay que darle prioridad a la hora de realizar el horario para el docente.

    IdDocente: Describe el identificador del docente, con este se conoce el docente al cual se le programara el horario.

14. **DocentesSeccion:** Esta tabla se construye con el fin de romper la relación muchos a muchos que existía entre la tabla Docentes y la tabla Sección. Esta tabla tiene los siguientes campos:

    IdDocente: Describe el identificador del docente, con este se conoce el docente que va a dictar la sección del curso.

    NRC: Describe el identificador de la sección, con este se conoce la sección del curso la cual será programada.

15. **Edificios:** En esta tabla se encuentra toda la información de los edificios que conforman la universidad. Esta tabla tiene los siguientes campos:

IdEdificio: Describe el identificador del edificio en el cual se programaran los cursos.

IdCampus: Describe el identificador del campus, con este se conoce el campus en el cual esta ubicado el edificio.

NombreEdificio: Describe el nombre del edificio en el cual se programaran los cursos.

SiglaEdificio: En este campo se establece las siglas del edificio en el cual se programaran los cursos.

16. **Franjas:** En esta tabla se indica toda la información de las franjas horarias en la que los docentes pueden dictar sus cursos. Esta tabla tiene los siguientes campos:

    IdFranja: Describe el identificador de las franjas horarias del docente en la cual el docente podrá impartir sus cursos.

    Descripción: Se establece las franjas horarias en los cuales el docente puede dictar sus cursos.

17. **FranjasPrograma:** Esta tabla se construye con el fin de romper la relación muchos a muchos que existía entre la tabla Franjas y la tabla Programa. Esta tabla tiene los siguientes campos:

    IdPrograma: Describe el identificador del programa, con este se conoce el programa al cual hay que darle prioridad a la hora de realizar el horario para el docente.

IdFranja: Describe el identificador de las franjas horarias, con este se conoce las franjas horarias del docente que este tiene para dictar el curso a programar.

18. **Periodo:** En esta tabla esta toda la información relacionada con el periodo el cual se va a programar. Esta tabla tiene los siguientes campos:

    IdPeriodo: Describe el identificador del periodo a ser programado.

    NombrePeriodo: Se establece el nombre del periodo a ser programado.

    InicioPeriodo: Se establece la fecha de inicio del periodo a programar.

    FinPeriodo: Se establece la fecha de finalización del periodo a ser programar.

19. **Programas:** En esta tabla se encuentra toda la información relacionada con los programas de la universidad. Esta tabla tiene los siguientes campos:

    IdPrograma: Describe el identificador del programa al cual se le programaran sus cursos.

    MAJR_CODE: En este campo se establece las siglas del programa al cual se le programaran sus cursos. Esto se realiza con el fin que el sistema de horarios sea compatible con campos específicos del sistema de matriculas BANNER.

    NombrePrograma: Se establece el nombre del programa al cual se le programaran sus cursos.

20. **Requisitos:** En esta tabla esta toda la información relacionada con los requisitos de los cursos a programar. Esta tabla tiene los siguientes campos:

    IdCursoPrograma: Describe el identificador del curso por programa, con este se conoce cual es el curso que necesita los requisitos para ser programado.

    IdRequisito: Describe el identificador del requisito, con este se conoce los requisitos del curso a ser programado.

21. **Reuniones:** En esta tabla se encuentra toda la información relacionada con las reuniones de cada sección de un curso a programar. Esta tabla tiene los siguientes campos:

    IdReunion: Describe el identificador de la reunión de cada sección de un curso a programar.

    IdPeriodo: Describe el identificador del periodo, con este se conoce en que periodo será programado la reunión de la sección del curso**.**

    NRC: Describe el identificador de la sección, con este se conoce la sección a la cual pertenece la reunión hacer programada.

    IdSalon: Describe el identificador del salón, con este se conoce el salón al cual se le asignara la reunión de la sección del curso a ser programado.

    IdDiaSemana: Describe el identificador del día de la semana en el cual será programada la reunión de la sección del curso.

HoraInicio: Se establece la hora de inicio de la sección que va hacer programada.

HoraFin: Se establece la hora final de cada reunión de la sección hacer programada.

DuracionMinutos: En este campo se establece la duración en minutos de la reunión de la sección que será programada.

22. **Salones:** En esta tabla se encuentra toda la información relacionada con los salones con los que cuenta la universidad. Esta tabla tiene los siguientes campos:

    IdSalon: Describe el identificador del salón.

    NombreSalon: Se establece el nombre del salón.

    IdEdificio: Describe el identificador del edificio, con este se conoce el edificio donde esta ubicado el salón.

    IdTipoSalon: Describe el identificador del tipo de salón, con este se conoce el tipo de salón que se utilizara, si es un salón practico o un salón teórico.

    Capacidad: se establece la capacidad del salón, con el fin de saber cuantos estudiantes puede recibir el salón hacer programado.

23. **SalonesPrograma:** Esta tabla se realiza con el fin de eliminar la relación muchos a muchos que existía entre la tabla salones y la tabla programas, Esta tabla tiene los siguientes campos:

    IdPrograma: Con este campo se establece el programa, con esto se conoce cual es el salón correcto que debe programarse para los cursos del programa.

    IdSalon: Se establece el salón en el cual serán programados los cursos del programa, con el fin de ubicar los cursos en el salón correspondiente, por ejemplo, un laboratorio quede en su respectivo curso, un curso de programación quede en su respectiva aula , etc.

24. **Secciones:** Esta tabla contiene toda la información relacionada con las secciones de cada curso a programar: Esta tabla tiene los siguientes campos:

    NRC: Describe el identificador de la sección.

    IdCursoPrograma: Describe el identificador del curso por programa, con este se conoce el curso por programa al cual se le programara sus secciones.

    IdDocente: Describe el identificador del docente, con este se conoce el docente que dictara la sección del curso a programar.

25. **Sesiones:** En esta tabla se encuentra toda la información de las sesiones de cada curso hacer programado, esta tabla contiene los siguientes campos:

Idseccion: Describe el curso hacer programado, con esto se conoce cual es el curso que se le programara la sesión.

IdSesion: Establece el identificador del sesión del curso hacer programada.

DuracionMinutos: indica la duración en minutos de la sesión hacer programada.

IdTipoSalon: En este campo se establece el tipo de salón en el cual será programada la sesión.

26. **SubFranjas:** En esta tabla se indica toda la información relacionada con las subfranjas. Esta tabla tiene los siguientes campos:

    IdSubFranja: Describe el identificador de la subfranja.

    IdFranja: Describe el identificador de la franja a la cual pertenece la subfranja.

    IdDiaSemana: Describe el identificador del día de la semana donde esta ubicada la subfranja.

    HoraInicio: Describe la hora de inicio de la subfranja.

    HoraFin: Describe la hora que finaliza la subfranja.

27. **TipoSalon:** En esta tabla se encuentra toda la información del tipo de salón que se va a utilizar. Esta tabla tiene los siguientes campos.

IdTipoSalon: Describe el identificador del tipo de salón, con este se conoce que tipo de salón se va a utilizar, si es un salón practico, teórico o un aula de informática.

Descripción: Se establece el nombre del tipo de salón, si es un salón práctico o un salón teórico.

28. **SubFranjasPosibles:** En esta tabla se encuentra toda la información relacionada con la población inicial que el algoritmo genético requiere para empezar a trabajar, esta información es la siguiente:

    IdHorario: Contiene la identificación única durante el proceso de evaluación de cada horario generado.

    Día: Como su nombre lo indica, es el día de la semana dentro del horario en el cuál serán programadas clases. El tipo de campo es numérico debido a que se realiza una numeración en orden ascendente de los días siendo de esta forma el lunes el día 1 y así sucesivamente.

    IdSalón: Cada salón disponible dentro de las instalaciones de la universidad destinadas para los programas de pregrado tratados por el software, se encuentra identificado con un número que lo representa en la base de datos del sistema.
    Hora Inicio/Fin: Representa las horas de inicio y fin en que será dictada una sección de un curso; en otras palabras una clase para el estudiante.

    IdProgramaResponsable: Indica la facultad o programa al cuál pertenece la sección del curso programado en el horario.

Clase: Es el semestre al que corresponde la sección programada.

IdCurso : identificador del curso al cual se le programarán las secciones.

IdDocente: Código con el que el docente es identificado en el sistema para llevar a cabo la programación de sus clases.

Idsección: Corresponde al identificador que se genera para cada grupo del curso a programar.

IdSesiónCurso: Son los bloques de horas en los que se programa dictar una sección de un curso.

29. **SubFranjasGenetico:** En esta tabla se encuentra toda la información relacionada con los horarios que han sido seleccionados para aplicarles los operadores genéticos de cruce y mutación, esta información es la siguiente:

    IdHorario: Contiene la identificación única durante el proceso de evaluación de cada horario generado.

    Día: Como su nombre lo indica, es el día de la semana dentro del horario en el cuál serán programadas clases. El tipo de campo es numérico debido a que se realiza una numeración en orden ascendente de los días siendo de esta forma el lunes el día 1 y así sucesivamente.

    IdSalón: Cada salón disponible dentro de las instalaciones de la universidad destinadas para los programas de pregrado tratados por el software, se encuentra identificado con un número que lo representa en la base de datos del sistema.

Hora Inicio/Fin: Representa las horas de inicio y fin en que será dictada una sección de un curso; en otras palabras una clase para el estudiante.

IdProgramaResponsable: Indica la facultad o programa al cuál pertenece la sección del curso programado en el horario.

Clase: Es el semestre al que corresponde la sección programada.

IdCurso : identificador del curso al cual se le programarán las secciones.

IdDocente: Código con el que el docente es identificado en el sistema para llevar a cabo la programación de sus clases.
Idsección: Corresponde al identificador que se genera para cada grupo del curso a programar.

IdSesiónCurso: Son los bloques de horas en los que se programa dictar una sección de un curso.

30. **Evaluación:** En esta tabla se encuentra toda la información relacionada con los valores de las restricciones utilizadas para trabajar con la función de evaluación, esta información es la siguiente:
IdHorario: Número del horario evaluado dentro de la generación.

CSMS (Cruce de Secciones del Mismo Semestre): El valor que contenga este campo representa el número de veces que se cruza una sección de un curso en un mismo semestre durante la semana planificada para las clases.

CSS (Cruce Secciones Semestre): El campo contiene el número de cruces de secciones sin el importar el semestre, solo teniendo en cuenta que coincidan la hora y el salón.

CD (Cruce de Docentes): Representa la cantidad de veces que durante la semana y en el horario generado un docente repite clase a la misma hora en un mismo día.

CSC (Cruce Sesiones Curso): Es el número de veces que una sesión de un curso se repita el mismo día.

BD (Bloque Día): La restricción BD contiene el promedio de horas diarias que se dictarán de una sección de un curso en el horario generado.

BC (Bloque Carrera): En este campo se almacena el promedio de horas de los bloques en cada horario tomando en cuenta todas las carreras y los semestres de cada una.

BDO (Bloque Docentes): Contiene el promedio de horas por bloque que un docente dicta en un día.

CSA (Cruce Salones): Representa el número de cruces de salones por día que en un mismo horario llegan a tener los semestres simultáneos.

Calificación: Almacena el valor de la función de evaluación aplicada al horario generado.

31. **MejoresGeneracion:** En esta tabla se almacena la evaluación de los mejores horarios de cada generación que fue ejecutada en el algoritmo genético, esta información es la siguiente:

    IdGeneracion: identificador de la generación al cual pertenece le horario seleccionado como el mejor de esta.

    IdHorario: Número del horario evaluado dentro de la generación.

    CSMS (Cruce de Secciones del Mismo Semestre): El valor que contenga este campo representa el número de veces que se cruza una sección de un curso en un mismo semestre durante la semana planificada para las clases.

    CSS (Cruce Secciones Semestre): El campo contiene el número de cruces de secciones sin el importar el semestre, solo teniendo en cuenta que coincidan la hora y el salón.

    CD (Cruce de Docentes): Representa la cantidad de veces que durante la semana y en el horario generado un docente repite clase a la misma hora en un mismo día.

    CSC (Cruce Sesiones Curso): Es el número de veces que una sesión de un curso se repita el mismo día.

    BD (Bloque Día): La restricción BD contiene el promedio de horas diarias que se dictarán de una sección de un curso en el horario generado.
    BC (Bloque Carrera): En este campo se almacena el promedio de horas de los bloques en cada horario tomando en cuenta todas las carreras y los semestres de cada una.

BDO (Bloque Docentes): Contiene el promedio de horas por bloque que un docente dicta en un día.

CSA (Cruce Salones): Representa el número de cruces de salones por día que en un mismo horario llegan a tener los semestres simultáneos.

Calificación: Almacena el valor de la función de evaluación aplicada al horario generado.

32. **MejoresHorarios:** Contiene toda la información del mejor horario que se encontró n todo el proceso del algoritmo genéticos, esta información es la siguiente:

    IdGeneracion: identificador de la generación al cual pertenece le horario seleccionado como el mejor de todo el proceso generado.

    IdHorario: Contiene la identificación única durante el proceso de evaluación de cada horario generado.

    Día: Como su nombre lo indica, es el día de la semana dentro del horario en el cuál serán programadas clases. El tipo de campo es numérico debido a que se realiza una numeración en orden ascendente de los días siendo de esta forma el lunes el día 1 y así sucesivamente.

    IdSalón: Cada salón disponible dentro de las instalaciones de la universidad destinadas para los programas de pregrado tratados por el software, se encuentra identificado con un número que lo representa en la base de datos del sistema.

Hora Inicio/Fin: Representa las horas de inicio y fin en que será dictada una sección de un curso; en otras palabras una clase para el estudiante.

IdProgramaResponsable: Indica la facultad o programa al cuál pertenece la sección del curso programado en el horario.

Clase: Es el semestre al que corresponde la sección programada.

IdCurso : identificador del curso al cual se le programarán las secciones.

IdDocente: Código con el que el docente es identificado en el sistema para llevar a cabo la programación de sus clases.
Idsección: Corresponde al identificador que se genera para cada grupo del curso a programar.

IdSesiónCurso: Son los bloques de horas en los que se programa dictar una sección de un curso.

8. DESARROLLO DEL ALGORITMO GENETICO

Los Algoritmos Genéticos simulan el proceso de evolución natural para Solución a problemas del mundo real. En este caso nuestro Algoritmo Genético dar solución al problema de la programación de horarios de la Universidad Autónoma de Bucaramanga.

Para que el Algoritmo Genético le de una buena solución a este problema de la Programación de los Horarios, este debe tener en cuanta lo siguiente:

8.1 POBLACIÓN INICIAL

La población inicial se genera con la siguiente formula:

$$PI = \#dias * \#horas\_dia * \#salones\_dia$$

Donde #dias es el número de días en los cuales se impartirán los cursos hacer programados, en la Universidad Autónoma de Bucaramanga se programan cursos de lunes a sábado.

#horas_dia corresponde al número de horas que tiene un día para programar los cursos, en la Universidad se programan cursos desde las 6:00 AM hasta las 10:00 PM sin excepción todos los días.

#Salones_dia corresponde al número de salones que hay en un día para programar los cursos.

En el software ésta población está representada por el número de horarios determinados por el usuario en la interfaz inicial del proceso. Está población inicial, se almacenará en la base de datos del sistema cuya tabla se llama SubFranjasPosibles la cuál presenta los siguientes campos de información:

IdHorario: Contiene la identificación única durante el proceso de evaluación de cada horario generado.

Día: Como su nombre lo indica, es el día de la semana dentro del horario en el cuál serán programadas clases. El tipo de campo es numérico debido a que se realiza una numeración en orden ascendente de los días siendo de esta forma el lunes el día 1 y así sucesivamente.

IdSalón: Cada salón disponible dentro de las instalaciones de la universidad destinadas para los programas de pregrado tratados por el software, se encuentra identificado con un número que lo representa en la base de datos del sistema.

Hora Inicio/Fin: Representa las horas de inicio y fin en que será dictada una sección de un curso; en otras palabras una clase para el estudiante.

IdProgramaResponsable: Indica la facultad o programa al cuál pertenece la sección del curso programado en el horario.

Clase: Es el semestre al que corresponde la sección programada.

IdCurso : Identificador del curso al cual se le programarán las secciones.

IdDocente: Código con el que el docente es identificado en el sistema para llevar a cabo la programación de sus clases.

Idsección: Corresponde al identificador que se genera para cada grupo del curso a programar.

IdSesiónCurso: Son los bloques de horas en los que se programa dictar una sección de un curso.

De acuerdo con la filosofía del algoritmo genético, esta población inicial será evaluada con los criterios que se han identificado durante el proceso de investigación para posteriormente utilizar los operadores genéticos (selección, cruce y mutación) que darán como resultado los mejores horarios de cada generación tomando en cuenta que el número de estas últimas, es ingresado por el usuario del sistema en la interfaz inicial del proceso.

8.2 FUNCIÓN DE EVALUACIÓN

Para poder evaluar una solución en un Algoritmo Genético se debe tener una función de evaluación, esta función de evaluación es la encargada de establecer si los horarios creados por el Algoritmo Genético son buenas soluciones o si se descartan o se trabaja más con ella.
Una función de evaluación debe tener ciertas restricciones para que esta trabaje correctamente.

En el caso de la programación de los horarios, la función de evaluación debe tener en cuenta las siguientes restricciones establecidas para darle una calificación a los horarios creador por el Algoritmo Genético y determinar si se sigue trabajando con esta solución o se descarta definitivamente:

- Sin cruces de docente, salón, cursos de un mismo nivel.
- Sin huecos entre cursos.
- Sin cruces de Cursos que sean correquisitos uno del otro
- Sin cruce de cursos de semestres adyacentes que no sean requisitos entre si.
- Los cursos de un determinado programa queden ubicadas en el edificio respectivo siempre y cuado haya disponibilidad.
- No existan más de una reunión de una misma sección de un curso en el mismo día.

Directamente durante el proceso de selección realizado por el software, la función de evaluación esta compuesta por un promedio o calificación otorgada al horario a partir de una operación matemática cuyas variables están representadas por las restricciones mencionadas anteriormente y un porcentaje que se le dio a cada una de ellas de acuerdo a la relevancia e importancia que tienen al momento de programar las clases de los docentes y programas. Una vez realizad la operación anterior, a cada horario se le almacenará en la base de datos del sistema la calificación que ha obtenido en su evaluación para posteriormente sacar un promedio de las calificaciones obtenidas por todos los horarios de la generación y con este realizar el proceso de selección; tomando en cuenta que solo se le aplicarán los operadores genéticos a aquellos horarios que se encuentren con una buena calificación con respecto al promedio.

La tabla que dentro del sistema contendrá los resultados y la información generada por las operaciones mencionadas anteriormente tiene por nombre Evaluación y en ella encontramos los siguientes campos:

IdHorario: Número del horario evaluado dentro de la generación.

CSMS (Cruce de Secciones del Mismo Semestre): El valor que contenga este campo representa el número de veces que se cruza una sección de un curso en un mismo semestre durante la semana planificada para las clases.

CSS (Cruce Secciones Semestre): El campo contiene el número de cruces de secciones sin el importar el semestre, solo teniendo en cuenta que coincidan la hora y el salón.

CD (Cruce de Docentes): Representa la cantidad de veces que durante la semana y en el horario generado un docente repite clase a la misma hora en un mismo día.

CSC (Cruce Sesiones Curso): Es el número de veces que una sesión de un curso se repita el mismo día.

BD (Bloque Día): La restricción BD contiene el promedio de horas diarias que se dictarán de una sección de un curso en el horario generado.

BC (Bloque Carrera): En este campo se almacena el promedio de horas de los bloques en cada horario tomando en cuenta todas las carreras y los semestres de cada una.

BDO (Bloque Docentes): Contiene el promedio de horas por bloque que un docente dicta en un día.

CSA (Cruce Salones): Representa el número de cruces de salones por día que en un mismo horario llegan a tener los semestres simultáneos.

Calificación: Almacena el valor de la función de evaluación aplicada al horario generado.

Figura 19 Tabla de Evaluación

	IdHorario	CSMS	CSS	CD	CSC	BD	BC	BDO	CSA	calificacion
▶	1	0	2	8	5	1.457209	2.2631612	2.1126015	10	2.6416485
	2	1	2	7	5	1.9251735	6.2604165	2.0568373	9	2.8121214
	3	4	2	3	2	2.022619	3.9384918	2.095238	8	1.9528174
	4	10	1	5	5	1.9556549	6.982068	2.0420635	10	3.2489896
	5	18	0	4	1	2.076543	4.482716	1.9845835	6	2.9771924
	6	16	2	9	6	2.1805556	6.892361	2.0130951	8	4.8043003
	7	13	2	11	4	1.5532408	2.3467593	2.0210316	9	4.2460523
*	0	0	0	0	0				0	

El porcentaje que le fue otorgado a cada una de las restricciones mencionadas anteriormente y trabajadas por la función de Evaluación fue sacado fundamentalmente de las investigaciones realizadas sobre el proceso manual que tradicionalmente se realiza para la generación de horarios. De acuerdo a ello se sacaron niveles de complejidad y valoraciones en cuanto a la importancia que tienen algunas de ellas sobre otras; es decir por ejemplo que cuando se desea cuadrar el horario para un docente es más relevante y colabora mas con el proceso encontrar que exista la menor cantidad de cruces en las horas que este último dicta que el cruce de los salones disponibles.

De acuerdo a ello para el trabajo con la función de evaluación, el orden de importancia de las restricciones quedó de la siguiente manera empezando por las más importantes y finalizando con aquellas de menor importancia pero con vital relevancia en la generación de los horarios:

CSS (Cruce Secciones Semestre), CSA (Cruce Salones), CD (Cruce de Docentes), CSC (Cruce Sesiones Curso), CSMS (Cruce de Secciones del Mismo Semestre), BD (Bloque Día), BC (Bloque Carrera), BDO (Bloque Docentes)

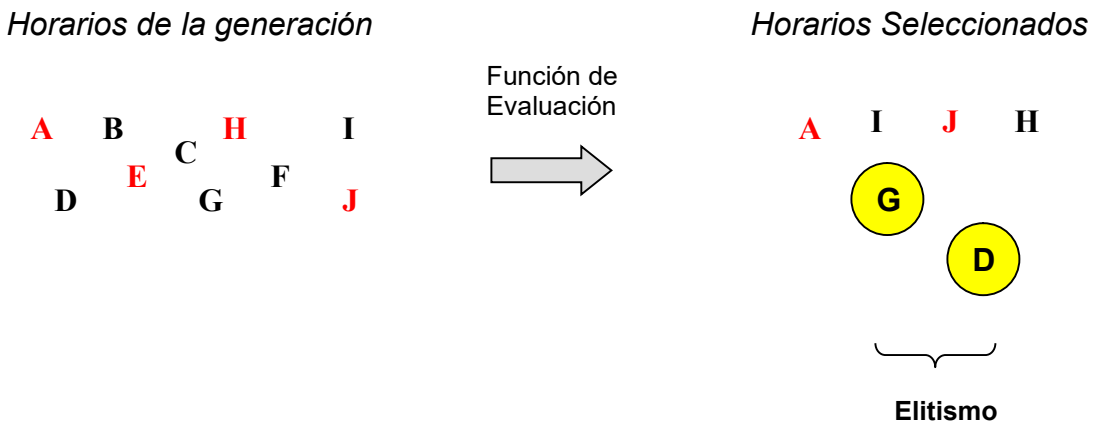
8.3 OPERADORES GENÉTICOS

8.3.1 Operador de Selección.

Según los criterios mencionados en la función de evaluación se procederá a seleccionar las soluciones que sean mas aptas para reproducirse con el fin de generar una nueva población que tenga una alta probabilidad de que en esta se encuentre la solución al problema planteado, en el caso de la programación de los horarios se procederá a seleccionar la solución que cumplan con las restricciones mencionadas anterior mente es decir se procederá a seleccionar las soluciones que obtengan las mejores calificaciones para así poder dar una solución mas rápida y optima al problema de la programación de horarios de la Universidad.

Como ya se había mencionado, esta selección se realiza tomando como base las restricciones y los valores que durante el análisis realizado toma cada una de ellas y que al final emergen en una calificación global para cada horario que en comparación con el promedio de la generación determinará su participación en las operaciones de cruce y mutación. Hasta el momento dentro del esquema planteado se había dejado a un lado un factor importante que también se toma en cuenta durante esta selección y que forma parte de los conceptos esenciales en el trabajo con algoritmos genéticos: “El Elitismo” que no es más que el paso a la siguiente generación de soluciones que tiene un porcentaje demasiado alto de eficiencia en la solución del problema. De esta manera, aquellos horarios que presenten estas características no participarán en operaciones de cruce ni mutación, sino, que serán seleccionados automáticamente para ser parte de la próxima generación sin ningún cambio. Esto significa que un horario en el que el nivel de cruces de salones, docentes, bloques, etc., es muy bajo en comparación a los demás horarios generados es un ejemplo claro de aquel que pueda ofrecernos una solución óptima y por consiguiente se puede aplicar el concepto de elitismo.

Figura 20 Selección de Horarios



8.3.2 Operador de Cruce

El operador de Cruce se utiliza con el fin de intercambiar ciertas partes del código genético de las soluciones que se han sido seleccionadas, esto se hace con el fin de que las nuevas soluciones hereden características de sus padres que a ellos los hicieron exitosos, para que estos nuevos individuos tengan una alta probabilidad de éxito y para que estos nuevos individuos le den una rápida solución al problema de la programación de horarios de la Universidad.

Como en el caso de la función de evaluación, el operador de Cruce debe tener en cuantas ciertas restricciones con el fin de cruzar los individuos correctamente. En el caso de la programación de horarios las restricciones que el operador de cruce debe tener en cuanta son las siguientes:

- Los cursos queden ubicados en el tipo de salón que les corresponde.
- Respetar la capacidad de cada salón.
- Respetar las asignaciones de salones definidas por la UNAB.
- Evitar romper reuniones.

- Intercambiar reuniones de la misma intensidad horaria y diferente cursos

El trabajo de cruce realizado por el software tiene en cuenta como se mencionaba, aquellos horarios que fueron seleccionados y que no hacen parte del proceso de elitismo. Es decir, aquellos horarios que tuvieron una buena calificación serán cruzados entre sí y teniendo en cuenta el concepto de cruce, para nuestro caso los cromosomas a intercambiar entre individuos estarán representados por los semestres de un mismo programa en horarios diferentes.

La selección de los semestres que serán cruzados entre los horarios que participarán en esta operación, se realiza a través de una mascara Binaria cuya longitud o número de posiciones estará determinada por la cantidad de semestres que hallan en el programa. Por ejemplo, el programa de Ingeniería de Sistemas consta de 10 semestres por lo cual si quisiéramos cruzar dos horarios de él, la máscara tendría 10 posiciones binarias y aquellas en donde aleatoriamente su contenido sea un 1 binario, representará los semestres a cruzar, es decir que si en la posición 4 de la máscara existe un 1 binario, se cruzará el semestre 4 de los dos horarios.

Figura 21 Cruce de Horarios

8.3.3 Operador de Mutación.

El operador de Mutación se utiliza con el fin de explorar regiones del dominio de problema que hasta el momento no se han visto, este operador también se utiliza para destruir los ciclos repetitivos que a veces produce el operador de cruce, pues en algunos casos el operador de cruce se queda esperando una mejor solución que no llegara, pues solo esta explorando la misma región del dominio del problema, en este momento entra en funcionamiento el operador de mutación para poder explorar las regiones del dominio del problema que el operador que cruce no a explorado hasta ese momento.
Cabe anotar que la probabilidad de cruce es mucho más alta que la probabilidad de mutación, pues el operador de cruce se utiliza más que el operador de mutación en un Algoritmo Genético.

Para poder Mutar una solución correctamente y poder explorara con éxito las regiones del dominio del problema que hasta el momento no se ha visto, el operador de Mutación debe respetar ciertas restricciones establecidas.

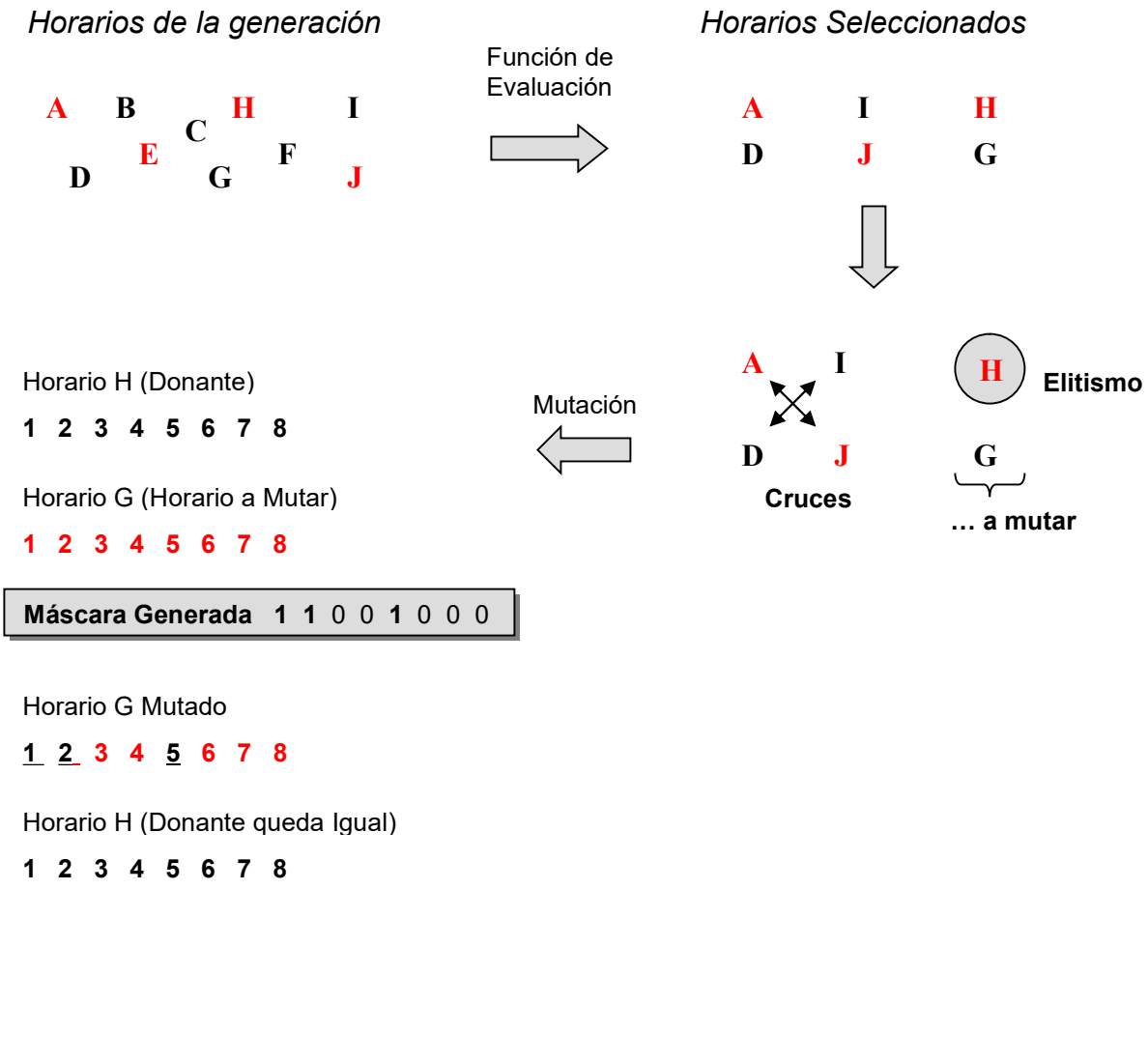
Para el caso de la programación de horarios de la Universidad el operador de Mutación debe respetar las mismas restricciones establecidas para el operador de Cruce. Estas restricciones son las siguientes:

- Los cursos queden ubicados en el tipo de salón que les corresponde.
- Respetar la capacidad de cada salón.
- Respetar las asignaciones de salones definidas por la UNAB.
- Evitar romper reuniones.
- Intercambiar reuniones de la misma intensidad horaria y diferente cursos

La mutación realizada por el software toma en cuenta aquellos horarios que fueron seleccionados y no hicieron parte de los cruces. Esta situación anterior se presenta debido a que en ocasiones después de realizado el proceso de elitismo con los horarios que obtuvieron buenos resultados en la función de evaluación, el número de estos últimos que se cruzará es impar y la misma operación nos lleva a que siempre quede un horario sin modificar genéticamente. Cuando se presenta un escenario como éste, al último horario se le aplica la operación de mutación la cual consiste en tomar uno de los horarios seleccionados preferiblemente un elitista para que le done algunos cromosomas definidos por una máscara al igual que se hizo en los cruces. El nuevo horario estará conformado entonces por los semestres donados que remplazarán a los anteriores y el resto que quedará igual; teniendo en cuenta que el donante no sufrirá ningún cambio por la misma función que cumple dentro de la operación de mutado. Por ejemplo, tenemos una generación en la que 6 horarios fueron seleccionados de acuerdo a la función de evaluación; uno de los cuales ofrece una solución muy cercana para el problema por lo cual pasa a la siguiente generación sin aplicarle los operadores genéticos (Elitismo).

De esta forma los restantes (5 Horarios) se cruzarán y de acuerdo a ello uno de estos sobrará en la operación al cual se le aplicará la mutación y para el caso el donante será el horario que por elitismo paso a la otra generación.

Figura 22 Mutación de Horarios



9. CONCLUSIONES

Se encontró que los algoritmos genéticos ofrecen una forma práctica y eficiente de mejorar el proceso de generación de Horarios para los diferentes programas ofrecidos por la Universidad Autónoma de Bucaramanga y a través de la investigación podemos asegurar que su filosofía de trabajo compacta idóneamente con las características de este proceso para llegar a soluciones óptimas.

Se estudiaron las bases y los conceptos fundamentales de los algoritmos genéticos para ser aplicados dentro del escenario planteado y se llegó a la conclusión de que tomando en cuenta variables importantes como las mismas restricciones ofrecidas por la generación de horarios, dichos conceptos pueden ser encaminados a apoyar este proceso a través de herramientas de programación como Java.

La estructura o la forma en que trabaja un algoritmo genético se amoldan de manera práctica a los mismos conceptos que se manejan en la elaboración o generación de horarios por parte de la Universidad.

Se encontró que la función de evaluación construida a partir de las restricciones planteadas por la generación de horarios presenta un carácter de flexibilidad que apoya el trabajo del algoritmo genético en la consecución de soluciones cercana. Esto último le permite al usuario del sistema variar el trabajo con ellas para alcanzar con mejores tiempos de respuesta.

REFERENCIAS

AGUILAR JOYANES LUIS, MARTINEZ ZAHONERO IGNACIO. “Programación en Java 2”

BOOCH GRADY, RUMBAUGH JAMES, JACOBSON IVAR. “El lenguaje unificado de modelado UML”.

PRESSMAN S, ROGER. “Ingeniería del software un enfoque practico cuarta edición”

SUN MICROSYSTEM. “Object- oriented Analysis and Design Using UML”.

[1] MARCOS AURELIO CARVALCAINT PACHECO
ICA: Núcleo de Pesquisa en Inteligencia Artificial
Pontificia Diversidad de Católica Rió de Janeiro
Algoritmos Genéticos: Principios y Aplicaciones
http://www.ica.ele.puc-rio.br/pesquisa/download/paper.pdf
Fecha: 19 de febrero del 2003

[2] PABLO ESTEVEZ VALENCIA
Instituto de Ingenieros de Chile
Optimización Mediante Algoritmos Genéticos
http://cipres.cec.uchile.cl/~em753/pdf/optimizacion.pdf
Fecha: 12 de Marzo del 2003

[3] SERGIO NESMACHOW

Universidad de la Republica, Instituto de Computo

Evolución en el diseño de los Algoritmos Genéticos Paralelos

http://www.fing.edu.uy/~sergion/gp/documentos/propios/EDCAGP.pdf

Fecha: 8 de Septiembre del 2005

[4] JAVA EN CASTELLANO

Tutoría de Java

http://programacion.com/java/tutorial/jdcbook/5/#jdcbookcalculodistribuido