HERRAMIENTA BASADA EN TÉCNICAS DE INTELIGENCIA ARTIFICIAL, PARA LA ASIGNACIÓN DE HORARIOS Y RECURSOS ACADÉMICOS, EN EL PROYECTO CURRICULAR DE INGENIERÍA DE SISTEMAS DE LA UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

WILMAR ALIRIO DÍAZ **20072020025**
CRISTIAN GUILLERMO GARCÍA **20072020030**
JIMMY ESTEBAN HERNÁNDEZ **20072020036**

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS
FACULTAD DE INGENIERÍA
PROYECTO CURRICULAR INGENIERÍA DE SISTEMAS
BOGOTÁ
2015

HERRAMIENTA BASADA EN TÉCNICAS DE INTELIGENCIA ARTIFICIAL, PARA LA ASIGNACIÓN DE HORARIOS Y RECURSOS ACADÉMICOS, EN EL PROYECTO CURRICULAR DE INGENIERÍA DE SISTEMAS DE LA UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

WILMAR ALIRIO DÍAZ **20072020025**
CRISTIAN GUILLERMO GARCÍA **20072020030**
JIMMY ESTEBAN HERNÁNDEZ **20072020036**

Proyecto de grado para optar al título de ingeniero de Sistemas

Director
LUZ DEICY ALVARADO NIETO
Ph.D en Ciencias de la Computación e Inteligencia Artificial

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS
FACULTAD DE INGENIERÍA
PROYECTO CURRICULAR INGENIERÍA DE SISTEMAS
BOGOTÁ
2015

Contenido

Tabla de Ilustraciones

Tabla de Ecuaciones

Tablas

1 INTRODUCCIÓN

El presente trabajo surge como solución al problema de asignación de espacios académicos en el Proyecto Curricular de Ingeniería de Sistemas de la Universidad Distrital Francisco José de Caldas, en donde los recursos académicos, horarios, asignaturas y docentes, son asignados mediante un proceso de prueba y error basándose en asignaciones anteriores, y que realizando cambios específicos, se ajusta a las particularidades del semestre en curso. Debido a la cantidad de variables y restricciones a tener en cuenta, dichas particularidades no siempre son satisfechas, y generan otros inconvenientes, como el desaprovechamiento del espacio físico, que los docentes deban someterse a los horarios ya existentes, o que el tiempo invertido por la persona encargada de dicha asignación sea considerablemente alto.

Este tipo de problemas (también conocidos como *timetabling*) han sido enmarcados dentro de los denominados NP-Duro (Györi, Petres, & Várkonyi-Kóczy, 2001), ya que además del gran número de variables que se deben tener en cuenta, el tiempo juega un papel importante en la construcción de una solución. La inteligencia artificial ha provisto un gran número de técnicas, herramientas y algoritmos para la solución del problema descrito ((Hernandez, Miranda, & Rey, 2008), (Saldaña Crovo, Oliva San Martín, & Pradenas Rojas, 2007)), ((Patat), (9th International Conference on the Pratice and Theory of Automated Timetabling, 2012)), ((NetWork, Metaheuristics, 2002), (Queen´s University Belfast, 2007), (CTIT Institute, 2011)); sin embargo, en el desarrollo realizado se utilizan tres de las técnicas más destacadas en la literatura: Búsqueda Tabú Adaptativa (Lü & Hao, 2008), Algoritmos Genéticos (Pérez de la Cruz & Ramírez Rodríguez, 2011) y Colonia de Hormigas (Socha, 2003).

Se presenta un modelo para la implementación de cada una de las técnicas al problema mencionado, así como una serie de pruebas realizadas para medir la calidad y el rendimiento de la solución generada, comparando los resultados con el método de asignación actual.

Como resultado del trabajo realizado se presenta un prototipo funcional que permite la creación automática de horarios, la edición manual de asignaciones realizadas, la visualización de horarios y restricciones violadas, y la generación y modificación de soluciones que contemplan carga académica, disponibilidad, recursos requeridos y perfil del docente, capacidad, disponibilidad, recursos y tipos de salón, espacios libres y jornadas continúas.

2 DESCRIPCIÓN DEL PROBLEMA

La Universidad Distrital Francisco José de Caldas se encuentra distribuida en varias facultades a lo largo de la ciudad de Bogotá. Cada una de las facultades cuenta con sus propios espacios físicos y dada la distancia entre las sedes, el uso de las aulas de otra facultad, genera dificultades en la movilización de estudiantes y docentes e influye en la programación de horarios, tal y como se presenta actualmente con asignaturas del Proyecto Curricular de Ingeniería de Sistemas que requieren de laboratorios de física ubicados en la sede de la Macarena.

En la Facultad de Ingeniería la distribución de los espacios físicos se realiza, en primera instancia, por parte de la oficina de planeación de la Universidad Distrital (Oficina de Planeacion, 2012). Partiendo de esta asignación, cada Proyecto Curricular hace su propia distribución de asignaturas, conforme a los diferentes espacios físicos disponibles y las particularidades propias del proyecto, tales como el tipo de vinculación de los docentes (Consejo Superior Universitario, 2002), su disponibilidad horaria y área de conocimiento. En el caso de Ingeniería de Sistemas, una vez que la oficina de planeación entrega la disposición de los salones para cada Proyecto Curricular, el coordinador toma como base las asignaciones de semestres anteriores y mediante negociaciones con profesores se realizan intercambios (cuando se hace necesario) que permitan satisfacer las necesidades del semestre en curso (cambios en la disponibilidad horaria de profesores, cambios en las franjas horarias de las materias, cambios en los grupos requeridos para cada asignatura, etc.).

En el proceso actual del proyecto curricular de Ingeniería de Sistemas, se genera un horario previo a la inscripción de asignaturas por parte de los estudiantes, pero posterior al proceso de matrícula. De este modo, una vez que el proyecto curricular tiene la estimación del número de estudiantes inscritos y la cantidad de grupos de cada asignatura que deben ser creados, se procede a la modificación del horario del semestre anterior.

La asignación manual toma bastante tiempo y es poco eficiente, dado que acarrea problemas como sobrecupos[1] , dificultades para realizar cambios y desaprovechamiento de espacios. De la mano de estos problemas surgen otros, como en el caso de los *video beams*, que deben ser transportados por los propios docentes, generando un deterioro de los equipos y que podría ser solucionado aprovechando los salones que cuentan con televisores previamente instalados.

Según la distribución entregada por el departamento de planeación (Oficina de Planeacion, 2012), existen 52 salones para la Facultad de Ingeniería, divididos entre los cinco proyectos curriculares de Ingeniería (Catastral y geodesia, Sistemas, Electrónica, Eléctrica e Industrial,) y posgrados. Los salones están disponibles para cada Proyecto Curricular en horarios específicos y donde solo 20 de ellos tienen la capacidad de

[1] Se hace referencia al hacinamiento que surge cuando se asigna un salón con una capacidad inferior al número de estudiantes inscritos a la asignatura

albergar más de 40 estudiantes. El Proyecto Curricular de Ingeniería de Sistemas utiliza 44 de los 52 salones existentes, siendo el segundo en hacer mayor uso de espacios físicos, sin tener en cuenta el uso de laboratorios[2]. Adicionalmente, el Proyecto Curricular de Ingeniería de Sistemas ha sido una de las carreras con mayor número de estudiantes de la facultad. Para el semestre 2012-3 aportó 1409 de los 6500 estudiantes activos, siendo la carrera con mayor número de estudiantes (ver Figura en la sección 10.1).

Se pretende entonces responder el siguiente interrogante: ¿Cuál de las técnicas de búsqueda: tabú, algoritmos genéticos o colonia de hormigas tendrá un mejor comportamiento en cuanto recursos, tiempo invertido y factibilidad de la solución generada, con respecto al proceso de asignación actual, en la programación de horarios, docentes, salones y asignaturas del Proyecto Curricular de Ingeniería de Sistemas de la Universidad Distrital Francisco José de Caldas?

3 MARCO TEÓRICO

3.1 CONTEXTUALIZACIÓN

Dentro del ámbito académico, se han establecido tres diferentes tipos de programación de horarios (Pitol Reyes, 2011) con respecto a las variables a tener en cuenta y la dificultad que estas representan para la solución del problema. El primero de ellos es la **asignación de horarios escolares,** en donde se tienen grupos o cursos de estudiantes fijos, que deberán ver una sesión en una franja de tiempo especificada. Es necesario evitar el cruce de profesores y de sesiones en una misma franja de tiempo, de modo que un profesor no podrá realizar 2 sesiones a una misma hora y que en un salón en una franja especificada no se dictará más de una asignatura.

El segundo tipo corresponde a la **asignación de horarios universitarios,** en donde se debe organizar un horario para las sesiones de un conjunto de asignaturas, teniendo en cuenta que, como tal, los estudiantes de un nivel académico no pertenecen a un curso fijo, y pueden inscribir asignaturas de niveles académicos diferentes al que están cursando. Además no todas las aulas cuentan con las características necesarias para una asignatura específica, y difieren en tamaño, obligando a revisar ¿qué aulas están disponibles, y para qué asignaturas? Es necesario tener en cuenta que las franjas horarias pueden cambiar y que la disponibilidad de los profesores varía y está ligada al tipo de vinculación que estos tienen con la institución.

El tercer y último tipo de *timetabling* ((MOSAIC Project Services Pty Ltd, 2006); (9th International Conference on the Pratice and Theory of Automated Timetabling, 2012); (Queen´s University Belfast, 2007); (NetWork, Metaheuristics, 2002); (CTIT Institute,

[2] El documento entregado por planeación solo presenta las cifras de los salones y no contempla ningún tipo de laboratorio.

2011); (Wren, 1996)) académico es el de **asignación de horarios de exámenes,** que consiste en asignar un horario para la realización de evaluaciones, teniendo en cuenta la disponibilidad de las aulas, la capacidad de las mismas y la duración del examen.

3.2 ANTECEDENTES

Los problemas de asignación de horarios en universidades, han sido objeto de varios estudios a nivel nacional e internacional. En los trabajos revisados que se presentan a continuación, se ha podido establecer una diferencia en los alcances de los mismos. Dicha diferencia consiste en que algunos de ellos, sólo han llegado a plantear un modelo de solución a través de alguna técnica, mientras que otros hacen un desarrollo a nivel de software, con el fin de realizar pruebas o de proveer una alternativa de solución según el enfoque especifico de cada trabajo.

A nivel internacional existen algunos estudios que tienen por objeto resolver problemas de tipo *timetabling* utilizando diversas técnicas de inteligencia artificial. Para el caso de algoritmos genéticos, es preciso resaltar tres trabajos diferentes realizados en México: El primero en la Universidad Autónoma de Zacatecas (Martínez Ruiz, Sánchez García, Muñoz Arteaga, & Castañeda Ramírez, 2006), donde se propuso un modelo basado en algoritmos genéticos y programación celular para resolver un problema de *timetabling*, sin realizar pruebas experimentales. El segundo tuvo lugar en 2010, en Guanajuato (Soria Alcaraz, Carpio, & Puga, 2010), donde se propuso un modelo que también se basa en algoritmos genéticos, pero con una metodología diferente llamada API-Carpio. El tercero y más reciente tuvo lugar en 2011 (Pérez de la Cruz & Ramírez Rodríguez, 2011), donde se plantea el problema de asignación de horarios como uno de coloración robusta, y se resuelve a través de una combinación entre un algoritmo genético y una búsqueda local.

Además de los algoritmos genéticos, también se han hecho desarrollos utilizando técnicas como colonia de hormigas o satisfacción de restricciones. Para el primer caso, en 2003, una variante del algoritmo de colonia de hormigas llamado Max-Min (Socha, 2003), surge a fin de resolver un problema de *timetabling*, Adicionalmente, de la mano de los algoritmos de colonia de hormigas, en 2009 se proponen 2 híbridos que permiten obtener soluciones más eficientes (Ayob & Jaradat, 2009). Uno de ellos, además de hacer uso de la colonia de hormigas, implementa un algoritmo de recocido simulado, mientras que el otro, realiza una combinación con un algoritmo de búsqueda tabú. Para el caso de satisfacción de restricciones, en 2005 en Australia, se propuso un modelo de satisfacción de restricciones para la programación de horarios en la Universidad de Wollongong, realizando las pruebas de dicho modelo en un software llamado ILOG (Zhang & Lau, 2005).

En Colombia también existen desarrollos relacionados con la programación de horarios. Dichos trabajos han sido realizados en diversas universidades y ciudades del país, y se caracterizan porque presentan el problema con un enfoque distinto (asignar horarios a

estudiantes) o precisan de particularidades diferentes a las del trabajo en cuestión. A continuación, una breve descripción de los trabajos realizados a nivel nacional.

En la universidad de los Andes se formuló un modelo de programación entera binaria (Castro Medina & Medaglia, 2004), en donde se definen "secciones" que corresponden a la dupla profesor - grupo de estudiantes, y "*slots*" como un conjunto de días de la semana, un patrón de tiempo y un salón. El modelo consiste en asignar una sección a un *slot*, realizando un pre procesamiento de los datos, y una vez generada una instancia reducida del problema, se procede a realizar la asignación de las secciones. En este trabajo se aplicaron 2 estrategias diferentes, la primera consistió en generar el horario de manera global, mientras que la segunda fraccionó el problema en asignaciones diarias.

En la sede Bogotá de la Universidad Nacional, se publicó un artículo donde se trabaja la programación de exámenes (Astaiza A., 2005). Aunque el tema central no es directamente la programación de horarios, el *scheduling* ((MOSAIC Project Services Pty Ltd, 2006); (Wren, 1996)) de los exámenes se asemeja bastante al de la asignación de espacios académicos. En este artículo se utiliza el método de coloreado de grafos para programar exámenes sin conflictos y se descompone en una serie de etapas. Dicho trabajo no presenta alguna aplicación para la solución de problemas de este tipo, pero sí provee un modelo a seguir a través de un algoritmo que contempla las variables grupos, asignaturas, salones y horarios.

La Universidad Distrital también ha abordado el tema de programación de asignaturas (Mendez Giraldo, Caballero Villalobos, & Álvarez Pomar, 2007). En 2007 se planteó un artículo donde, inicialmente, se propone dar solución al problema a través de un modelo de programación mixta, que no resultó exitoso desde el punto de vista práctico, obligando a que se propusiera un método combinado de las técnicas de algoritmos genéticos y búsqueda tabú. Este artículo tomó un enfoque algo diferente, ya que se pretendía crear también los horarios de los estudiantes teniendo en cuenta cierto coeficiente de peso que indicaría el grado de nivelación del estudiante, por lo que primó la satisfacción al estudiantado. Además, se hizo diferenciación entre las materias correspondientes a cada semestre y las materias electivas, lo que generó otras restricciones en el modelo.

En 2008, en la Universidad Tecnológica de Pereira se aborda el problema de programación óptima de horarios de clase, a través de la metaheurística colonia de hormigas (Peñuela, Franco B., & Toro O., 2008). Se plantean algunas matrices que representan la relación entre dos o más recursos a asignar (asignatura-salon, estudiante-asignatura, etc), y se resuelve a través de 3 fases: La primera, consiste en la asignación de los salones, dando prioridad a aquellos con menos opciones factibles; La segunda en la asignación de los bloques de tiempo para los diferentes eventos que se dictarán en el mismo salón; la tercera, representa el corazón del algoritmo de colonia de hormigas, donde se trata la evaporación y el depósito de las feromonas, teniendo en cuenta que, una vez hecha la asignación, se reduce la cantidad de feromonas que representa al evento asignado. Además de la creación del modelo, para la realización de las pruebas

se desarrolló una aplicación en lenguaje Delphi tomando datos del *International Timetabling Competition* (Queen´s University Belfast, 2007). Es necesario resaltar, que se hizo énfasis en trabajar de la mano de otro algoritmo como el recocido simulado ((Abramson, 1991), (Aycan & Ayav, 2009)), para generar soluciones iniciales y mejorar la convergencia de la colonia de hormigas.

Finalmente, ese mismo año en la universidad del Norte en Barranquilla, se planteó un algoritmo en la solución de un problema que contempla la capacidad del salón para albergar a los estudiantes (Caballero Mejia, 2008), sin tener en cuenta otras características adicionales.

A pesar del gran número de investigaciones y trabajos realizados en el área de la asignación de horarios para universidades, no se encontró ningún desarrollo que contemplara variables como recursos (de salones y requeridos por un docente), horarios de salones, capacidad de salón, número de estudiantes por grupo, asignaciones en bloque y en una misma jornada, etc... Si bien la mayoría de los trabajos tienen algunas características en común, ninguno contempla todas las variables trabajadas en el presente desarrollo. En la sección de resultados se presenta la Tabla 21 con una comparativa de los trabajos mencionados, frente a las restricciones contempladas en el presente trabajo. La letra P indica que la restricción es tenida en cuenta a través de datos predefinidos.

3.3 TÉCNICAS DE INTELIGENCIA ARTIFICIAL

Las técnicas de inteligencia artificial son un elemento clave para la resolución del problema que se aborda en este documento ((Lien-Fu, Nien-Li, Liang-Tsung, & Tien-Chun, 2006), (Peñuela, Franco B., & Toro O., 2008), (Györi, Petres, & Várkonyi-Kóczy, 2001)), ya que sus procesos algorítmicos proporcionan una solución considerablemente rápida. Dado que este tipo de algoritmos no contemplan la totalidad de las soluciones, requieren menor capacidad computacional que los algoritmos de fuerza bruta, y con soluciones aceptables.

A continuación se presentan los algoritmos mencionados frecuentemente en la literatura, y con resultados prometedores para el tratamiento del problema de asignación de horarios y espacios académicos. Se hará uso de la siguiente notación para la representación en pseucódigo de cada uno de ellos:

S: Solución

S_0: Solución inicial

S_C: Solución Candidata

S_{mejor}: Mejor solución

L_C: Lista de soluciones candidatas

Fo(S): Función objetivo de la solución S

3.3.1 ALGORITMO VORAZ

El nombre de algoritmo voraz, también conocido como ávido (su nombre original proviene del término inglés *Greedy*) se debe a su comportamiento: en cada etapa "toman lo que pueden" sin analizar consecuencias, es decir, son glotones por naturaleza.

El algoritmo voraz o "Greedy" es uno de los enfoques más simples para resolver problemas de optimización en los que se quiera determinar el óptimo global de una función dada. Estos algoritmos intentan construir una óptimo local buscando paso a paso, eligiendo en cada uno de ellos el componente de la solución que parece más apropiado (Muniswamy, 2009).

Este tipo de algoritmos nunca revisan una elección ya realizada, confiando en haber elegido bien los componentes previos. Razón por la que no siempre dan buen resultado.

Para que un problema sea susceptible de ser solucionado por un algoritmo voraz, se deben definir una serie de elementos que han de estar presentes:

- Un conjunto de candidatos que corresponde a las n entradas del problema.

- Una función de selección que en cada momento determine el candidato idóneo para formar la solución de entre los que aún no han sido seleccionados ni rechazados.
- Una función que compruebe si un cierto subconjunto de candidatos es prometedor. Se entiende por prometedor, que sea posible seguir añadiendo candidatos y encontrar una solución.
- Una función objetivo que determine el valor de la solución hallada. Es la función que se quiere maximizar o minimizar.
- Una función que compruebe si un subconjunto de estas entradas es solución al problema, sea optimo o no.

Con estos elementos, se puede resumir el funcionamiento del algoritmo voraz en los siguientes puntos:

1. Para resolver el problema, un algoritmo voraz tratará de encontrar un subconjunto de candidatos tales que, cumpliendo las restricciones del problema, constituya la solución óptima.
2. Trabajará en etapas, tomando en cada una de ellas la decisión que parezca mejor sin considerar las consecuencias futuras. Por tanto, escogerá de entre todos los candidatos el que produce un óptimo local para esa etapa, suponiendo que será a su vez óptimo global para el problema.
3. Antes de añadir un candidato a la solución que se está construyendo, comprobará si es prometedora al añadirlo. En caso afirmativo lo incluirá en ella y en caso contrario descartará este candidato para siempre y no volverá a considerarlo.
4. Cada vez que se incluye un candidato, comprobará si el conjunto obtenido es solución.

Pseudocódigo

```
Inicio AlgoritmoGreedy
    S ← 0
    Generar Lista de Candidatos Lx
    Mientras (Lx no sea vacía o no se cumpla una condición de salida)
        X ← Selección de Lx  //Seleccionar un elemento de Lx
        Lx ← Lx - X
        Si S U X es solución
                S ← S U X
    Terminar Mientras
    Retornar S.
Terminar Greedy
```

Ilustración 1: *Pseudocódigo del algoritmo Greedy (Muniswamy, 2009)*

A diferencia de otros métodos de optimización, los algoritmos voraces no siempre proporcionan una solución, debido a su naturaleza y dependiendo del problema en el cual se aplique, es muy probable que se estanquen en un óptimo local. Por esta razón, no todos los problemas son resolubles utilizando dicha metodología.

Algunos de los elementos que caracterizan a un algoritmo voraz son:

- No garantizan una solución óptima.
- Suelen ser rápidos y fáciles de implementar.
- Solo generan una entre todas las posibles secuencias de decisiones.
- Siempre habrá que demostrar que la solución obtenida es óptima.

Algunos problemas de aplicación que son más comúnmente usados para dar una explicación sobre el funcionamiento de los algoritmos voraces son (Jaramillo Villegas, Analisis y Diseño de Algoritmos, 2014):

- El problema de la devolución del cambio.
- El problema de la mochila discreta.
- El problema del agente viajero.

3.3.2 BÚSQUEDA TABÚ

Búsqueda Tabú o *Tabu Search (TS), a*parece por primera vez en 1986 cuando Fred Glover introduce el término "metaheurística" (Glover, 1986). Posteriormente, con la publicación de los fundamentos de búsqueda tabú (Glover & Laguna, 1997), surgen distintas aplicaciones que promueven el éxito de la técnica en la solución de problemas de optimización duros, destacando como una de las mejores en el área de la programación de horarios (Glover, Laguna, & Martí, 2007).

TS implementa estructuras de memoria que permiten guiar una búsqueda fuera de los óptimos locales, con la capacidad de contemplar otras soluciones en el espacio de búsqueda global. Esta estructura de memoria, conocida como lista tabú, contiene las soluciones que han sido evaluadas o los movimientos realizados para alcanzarla. Durante cada iteración se revisa la lista tabú y se dejan de lado aquellas alternativas registradas, reduciendo el número de posibilidades elegibles.

Con la implementación de esta técnica de optimización, aparecen 2 conceptos que la ubican dentro del campo de la inteligencia artificial: "memoria adaptativa" y "exploración responsiva". La primera surge debido a que las decisiones de búsqueda son influenciadas por la información que se ha obtenido a lo largo del proceso, contrastando con diseños semialeatorios y permitiendo implementar procesos de búsqueda eficaz y eficiente. Por otra parte, la exploración responsiva se deriva de la idea de no contemplar únicamente las soluciones más prometedoras, sino considerar las malas elecciones como fuentes de información. Al considerar los conceptos

mencionados anteriormente, TS se basa en estrategias derivadas de procesos del aprendizaje (Melián Batista & Glover, 2003) .

Búsqueda tabú
$S \leftarrow S_0$
Smejor ← S
Lista Tabu T← null
Mientras (no se alcance la condición de parada)
    L_C← null
    Por cada $S_c \in V$
    Si ($S_C \notin T \vee (S_C \in T \wedge$ Supera el criterio de aspiración))
        $L_C \leftarrow L_C + S_C$
    $S_C \leftarrow$ S con mejor Fo(S) en L_C
    Si Fo(S_C) es mejor que Fo(Smejor)
        $S_{mejor} \leftarrow S_C$
        $T \leftarrow T + S_C$
        Si | T | > máximo tamaño de T
            T ← T – S más antiguo
Devolver S_{mejor}
Terminar Búsqueda tabú

Ilustración 2: *Pseudocódigo Búsqueda Tabú .Basado en (Brownlee, 2014)*

3.3.2.1 *Uso de Memoria*

En TS se trabajan 4 características referentes al uso de la memoria:

- **Reciente**: Es posible conocer el tiempo transcurrido en que se ha contemplado una solución específica, permitiendo hacer un seguimiento al proceso de búsqueda.
- **Frecuencia:** Indica con qué periodicidad se contempla una solución. Permite establecer también el grado de calidad de la misma, ya que si se revisa de manera frecuente una alternativa, o un paso para alcanzar la solución, es probable que esta pertenezca al conjunto de los óptimos.
- **Calidad:** Permite establecer la bondad de una solución contemplada.
- **Influencia:** Indica el impacto de realizar cierto movimiento dentro del proceso de búsqueda.

Contemplando estas características, y a fin de garantizar las tareas de intensificación y diversificación, se hace uso de 2 tipos de memoria adaptativa:

Memoria a corto plazo: guarda información que permite guiar la búsqueda de forma inmediata, y desde el comienzo del procedimiento.

Memoria a largo plazo: guarda información que permite guiar la búsqueda a posteriori, después de una primera etapa en la que se han realizado una o varias ejecuciones del algoritmo aplicando la memoria a corto plazo

3.3.2.2 *Intensificación y diversificación*

Una de las características más importantes de búsqueda tabú, es su capacidad de salir de óptimos locales. Para lograr esto, se implementan las estrategias de intensificación y diversificación. En la primera, se recurre a la modificación de las reglas de selección, con el fin de favorecer las elecciones con mayor calidad o las mejores combinaciones de movimientos. Se comparan entonces la función objetivo y la mejor solución obtenida hasta el momento, con el fin de construir un conjunto de soluciones élite que permitirán dicha modificación.

En contraparte, las estrategias de diversificación permiten contemplar soluciones significativamente diferentes a las ya evaluadas, llevando el algoritmo a zonas del espacio de búsqueda que no han sido visitadas.

3.3.2.3 *Niveles de aspiración*

Los niveles de aspiración permiten que el proceso de búsqueda sea más flexible. Estos niveles permiten determinar cuándo es posible reemplazar restricciones tabú o eliminar la clasificación de tabú a un elemento dado.

Para manejar los niveles de aspiración se utilizan diversos criterios que pueden mejorar la búsqueda. Uno de los más usados es el criterio de calidad, que reemplaza un movimiento si conduce a una mejor solución que la obtenida hasta ahora. También es posible utilizar el criterio de influencia, midiendo el grado de cambio al realizar una acción. Al implementar este criterio, no se llevará a cabo un movimiento con la misma calidad de otro que ya se realizó, reduciendo el uso de acciones innecesarias en el proceso de búsqueda.

3.3.2.4 *Búsqueda Tabú Adaptativa o ATS* ***(Lü & Hao, 2008)***

Presentada en el ITC de 2007 como una de las estrategias de optimización con mejores resultados, quedando en segundo lugar de la competencia. Plantea el uso de 3 técnicas conocidas en la literatura a través de 3 fases claramente definidas: Inicialización, intensificación y diversificación.

Primera Fase (Inicialización):

Busca la creación de una solución inicial que sea factible pero que se construya rápidamente. Para alcanzar dicho objetivo, se hace uso de una heurística *voraz* rápida que se concentra en garantizar las restricciones duras y dejar en segundo plano el análisis de una solución óptima respecto a las restricciones blandas.

Segunda Fase (Intensificación):

Recurre al modelo general de *búsqueda tabú* seleccionando candidatos que no están en la lista tabú, o que si lo están cumplen con el criterio de aspiración. Agrega un criterio de parada denominado "Profundidad", que básicamente limita la cantidad de movimientos realizados en la búsqueda.

Tercera Fase (Diversificación):

Hace uso de la *búsqueda local iterativa* (Lourenco, Martin, & Stützle, 2003) para sacar el algoritmo de los óptimos locales. Cuando la segunda fase se queda estancada, la tercera se encarga de crear una nueva solución para realizar nuevamente la búsqueda por intensificación.

Para generar la solución requerida por la segunda fase, se realiza una perturbación (intercambio de movimientos sobre la solución) que genera el punto de inicio para una búsqueda local simple y, una vez se genera una solución factible, se entrega al algoritmo de búsqueda tabú. En esta fase, aparece la segunda clave de ATS denominada la "fuerza de perturbación", que representa la cantidad de perturbaciones generadas para guiar al algoritmo.

En ATS, los valores "profundidad" y "fuerza de perturbación" son modificados dinámicamente en el proceso para permitir la adaptación del algoritmo conforme a las soluciones que se van encontrando.

3.3.3 ALGORITMOS GENÉTICOS

En la universidad de Michigan en 1975 Jonh Holland (Holland, 1992), creó los algoritmos genéticos con el propósito de estudiar los fenómenos adaptativos de la naturaleza, haciendo una abstracción de la selección natural (Melanie, 1998). Los algoritmos genéticos son un tipo de algoritmos evolutivos (Chacón Montes, 1995), están compuestos por una función de aptitud o "*fitness*" y una población de cromosomas. Donde cada cromosoma se conforma por un número de genes, que corresponden a algún aspecto de la solución.

3.3.3.1 *Representación de los cromosomas*

Los cromosomas influyen en el rendimiento del algoritmo al momento de generar una solución, por eso según el problema abordado, pueden tener diversas representaciones (números reales, letras, objetos, números binarios, etc.).

3.3.3.2 *Operadores*

La forma típica de esta técnica hace uso de tres operadores: selección, cruce y mutación. (Melanie, 1998).

3.3.3.2.1 Selección

Elige los cromosomas usados en el operador de cruce y los que van a formar la población de la siguiente generación. Existen diferentes técnicas para hacer esta selección: elitista, proporcional a la aptitud, rueda de ruleta, escalada, por torneo, por rango, generacional, por estado estacionario, jerárquica, etc. ((Melanie, 1998), (Chacón Montes, 1995)).

3.3.3.2.2 Cruce

Este operador imita la recombinación entre dos cromosomas haploides. De forma aleatoria se elige el o los puntos de corte, definiendo el material genético que será intercambiado entre el par de cromosomas ((Melanie, 1998); (Chacón Montes, 1995)). En la literatura las técnicas de cruce más usadas son:

- Un punto de cruce: Se selecciona de forma aleatoria un punto para hacer el intercambio de genes entre los padres y los descendientes. (Wang, 2004).
- Dos puntos de cruce: Se selecciona de forma aleatoria dos puntos para hacer el intercambio de genes entre los padres y los descendientes. (Wang, 2004).
- Cruce uniforme: Mediante la asignación de una probabilidad, se determina el material genético que se dona a un descendiente. Este operador está sujeto a una probabilidad de ocurrencia (Pc), dándole a cada individuo la oportunidad de transmitir sus genes sin ser intercambiados; típicamente la probabilidad se encuentra en el rango de 0.6 y 1.0. (Wang, 2004).

3.3.3.2.3 Mutación

Opera sobre algunos genes cambiándolos de forma aleatoria y utilizando la evaluación según la función objetivo ((Melanie, 1998), (Chacón Montes, 1995)). Está sujeto a una probabilidad pequeña (Pm) que con frecuencia es de 0.001 (Wang, 2004), dándole a cada espacio de búsqueda una probabilidad diferente de cero para ser examinado. En la Ilustración 6 el punto de mutación está demarcado en rojo.

Cada iteración del algoritmo produce una nueva generación de posibles soluciones. En la primera se crea la población de forma aleatoria, luego se evalúa mediante la función de aptitud y se hace la selección de la nueva generación, buscando la supervivencia de los mejores individuos. Luego a los cromosomas seleccionados, se les manipula por medio de los operadores de cruce y/o mutación, dejando lista una nueva población mejorada para la siguiente iteración. El algoritmo se detiene al llegar a la solución o al número de ciclos máximo ((Rios, 2013), (Melanie, 1998)).

La estructura general del algoritmo genético según (Grefenstette & Baker, 1989) se describe de la siguiente forma:

Algoritmo Genético

$L_c \leftarrow$ *poblacioInicial()*
Evaluar función objetivo de cada elemento en L_c
Mientras (no se alcance la condición de parada) hacer
$L_{cPadres} \leftarrow$ *selecionarPadres* (L_c)
$L_{cCruce} \leftarrow$ *cruzarPadres* ($L_{cPadres}$)
Si condicionMutacionSatifecha entonces
aplicarM*utacion* (L_{cCruce})
Fin si
Evaluar función objetivo de cada elemento en L_{cCruce}
$L_c \leftarrow$ *construirSiguienteGeneracion*(L_{cCruce}, L_c)

Fin mientras
$S_{mejor} \leftarrow$ *mejorIndividuo* (L_c)
retornar S_{mejor}
Fin algoritmo

Ilustración 3: *Estructura general del algoritmo genético (Soria Alcaraz, Carpio, & Puga, 2010)*

3.3.4 COLONIA DE HORMIGAS

El algoritmo basado en Colonias de Hormigas o ACO por sus siglas en inglés *Ant Colony Optimization,* es una metodología de optimización en la cual se imita el comportamiento de las hormigas, quienes de manera muy eficiente consiguen alimento y se comunican. Este algoritmo se encuentra enmarcado dentro de la meta-heurística del *swarm intelligence* (Ayob & Jaradat, 2009).

La teoría de optimización por medio de algoritmos basados en colonias de hormigas fue presentada por primera vez por Dorigo y Di Caro en 1999 (Dorigo & Di Caro, 1999). La base de esta combinación de Sistemas (natural y artificial) es la comunicación indirecta entre todos los individuos a partir de rastros de feromonas, con lo cual se logran resolver problemas de optimización discreta (problema del agente viajero, camino más corto, programación dinámica, *sheduling*, etc.). El primer ejemplo en el cual se aplicó este algoritmo fue en el típico problema del viajante por el ingeniero italiano Marco Dorigo en 1992 (Dorigo M. , 1992).

El algoritmo ACO es de tipo iterativo, por lo tanto en cada iteración se "inicia" una colonia *m* de hormigas, cada una de las cuales encuentra una solución al problema de manera probabilística, guiándose por un rastro de feromonas y por una información calculada previamente de manera heurística.

Los valores de las feromonas son actualizados por todas las *m* hormigas que han construido una solución en la iteración por sí mismas. La feromona τ_{ij}, asociada con la arista i y j es actualizada de la siguiente manera:

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + \sum_{k=1}^{m}\Delta\tau_{ij}^{k}$$

Ecuación 1: *Ecuación que actualiza el rastro de las feromonas.* (Dorigo, Birattari, & Stutzle, Ant Colony Optimization: Artificial Ants as a Computational Intelligence Technique, 2006).

Donde ρ es la tasa de evaporación, m es el número de hormigas y $\Delta\tau^{k}_{ij}$ es la cantidad de feromona dejada en la arista i, j por la hormiga k.

$$\Delta\tau_{ij}^{k} = \begin{cases} Q/L_k & \text{Si la hormiga k usa la arista i, j en su viaje.} \\ 0 & \text{De otra forma} \end{cases}$$

Ecuación 2: *Cantidad de feromonas dejada en la arista i, j por la hormiga k.*

Q es una constante y L_k es la longitud del camino construido por la hormiga k.

En la construcción de la solución, las hormigas seleccionan el siguiente vértice a ser visitado a través de un mecanismo estocástico. Donde la hormiga k se encuentra en el

vértice i y ha construido la solución parcial S^p, la probabilidad de ir al vértice j está dada por:

$$p_{ij}^k = \begin{cases} \dfrac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{c_{il} \in \mathbf{N}(s^p)} \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}} & \text{Si } C_{ij} \text{pertenece a } N(s^p). \\ 0 & \text{De otra forma.} \end{cases}$$

Ecuación 3: *Probabilidad de la hormiga k de ir del vértice i al vértice j.*

Donde N(s) es el conjunto de componentes viables, es decir, las aristas (i, l), donde l es un vértice que aún no ha sido visitado por la hormiga k. Los parámetros α y β controlan la importancia relativa de la feromona contra la información de la heurística η_{ij}, que está dada por:

$$\eta_{ij} = 1/d_{ij}$$

Ecuación 4: *Información de la heurística.*

Siendo d_{ij} es la distancia entre los vértices i, j (Dorigo, Birattari, & Stutzle, 2006).

El algoritmo se detendrá después de que se cumpla la condición de parada o hasta cumplir un número de iteraciones determinadas.

Algoritmo Hormigas
- H← Número de Hormigas
- $S \leftarrow S_0$
- $S_{mejor} \leftarrow S$
- $L_C \leftarrow$ null
- Feromona F ← Tope Max F
- Mientras (no se alcance la condición de parada)
  - Por cada H hacer
    - $S_C \leftarrow S_H$ //Solución de una hormiga
    - $F_o(S_C)$ //Hallar Valor de la función Objetivo de la solución actual
    - Si $Fo(S_C)$ es mejor que $Fo(S_{mejor})$
      - $S_{mejor} \leftarrow S_C$
      - $F \leftarrow F + F_{inc}$ //Incremento Feromona
    - Si no
      - $F \leftarrow F + F_{dec}$ //Decremento Feromona
- Devolver S_{mejor}

Fin algoritmo

Ilustración 4*: Estructura general del algoritmo hormigas. Fuente Propia*

3.3.4.1 *MIN – MAX Ant System (Variante Min – Max de Colonia de Hormigas)*

El sistema de Hormigas Min – Max es una modificación del método de optimización propuesto por Dorigo y Di Caro el cual toma como base las ideas de Colonia de Hormigas. Este algoritmo fue propuesto por Stutzle y Hoos (Stutzle & Holger H., 2000), quienes introdujeron algunos cambios importantes, entre los cuales se encuentran:

- Tan solo la mejor hormiga es utilizada para actualizar los rastros de la feromona, esta es la que generó el mejor camino de la iteración actual o la que obtuvo el mejor tour desde el inicio del algoritmo.
- Limita el rango posible de los valores de los rastros de la feromona a un intervalo acotado entre los valores T max y T min, de tal manera que se contrarreste el efecto producido al permitir que solo la mejor hormiga actualice la feromona.
- En la primera fase del algoritmo, los rastros de las feromonas son inicializados en el valor límite T max, el cual junto al pequeño factor de evaporación, incrementa la exploración de los tour al comienzo de la búsqueda.
- Los rastros de la feromona son reinicializados cada vez que el sistema conduce a una situación de estancamiento o cuando después de varias iteraciones no ha sido generado ningún mejor tour (Arito, Leguizamón, & Errecalde, 2010).

3.3.4.1.1 Mecanismos de Exploración

Para el caso de los algoritmos basados en Colonia de Hormigas, es de gran ayuda utilizar los mecanismos de exploración, dado que permiten elegir en ciertos casos hasta el elemento menos deseable de un conjunto de soluciones (debido a sus características). Algunos de los métodos de exploración más conocidos son:

3.3.4.1.1.1 Método Aleatorio

En esta metodología no se tiene en cuenta información de ningún tipo, por lo que todos los elementos que pertenecen a un espacio solución tienen la misma probabilidad de ser escogidos como una posible solución.

3.3.4.1.1.2 Método Ruleta

Es un mecanismo en el cual se incorpora la exploración y la explotación, ya que si bien los elementos pertenecientes al conjunto solución que tengan una probabilidad baja pueden ser escogidos, los elementos mejor calificados o que posean la mejor probabilidad (mayor cantidad de feromona para este caso) resultan ser favorecidos.

A manera de ejemplo se podría pensar en 3 caminos posibles que puede tomar una hormiga, dentro de los cuales se encuentran relacionadas las probabilidades: 0,5; 0,2 y 0,3 para los caminos 1, 2 y 3 respectivamente. De tal manera que si se tuviera una ruleta con 100 posiciones y se aterrizaran estas probabilidades a la metodología, 50 de esas posiciones las ocuparía el camino 1, 20 de esas posiciones las ocuparía el camino 2 y por ultimo 30 de esas posiciones las ocuparía el camino 3. De tal manera que al "echar a rodar la bolita" el camino con mayor probabilidad para ser seleccionado como una solución, sería el camino 1.

3.3.4.1.1.3 Método Torneo

Este mecanismo se podría considerar como un híbrido entre el aleatorio y de la ruleta, ya que inicialmente todos los elementos del conjunto solución poseen la misma probabilidad de ser elegidos. Posteriormente se seleccionan tan solo dos de ellos para competir en un torneo en el que gana aquel que tenga las mejores características (Gómez Toro, Vanegas Castellanos, & Zuluaga Gómez, 2009).

4 MODELAMIENTO DE LA SOLUCIÓN

El presente desarrollo aborda el problema descrito a través de un software que considera aspectos no contemplados en modelos anteriores de programación de horarios y asignación de espacios, identificando aquellos aspectos que hacen parte del proceso manual actualmente en uso, modelándolos como variables y restricciones a cumplir. Se da prioridad a la disponibilidad horaria de los docentes, los recursos pedagógicos requeridos y la construcción de un horario que permita que los estudiantes puedan tomar cada una de las asignaturas del semestre al que se encuentran inscritos, tal y como sucede en el proceso actual.

Teniendo en cuenta cada uno de los aspectos mencionados, la implementación realizada toma en cuenta las siguientes variables (se explican con mayor detalle en el cálculo de la función objetivo):

Variables relacionadas con el salón:

- Horas en las que está disponible el salón: Según planeación (Oficina de Planeacion, 2012), un salón está disponible para cada Proyecto Curricular en ciertas franjas horarias.
- Capacidad del salón: Cantidad máxima de estudiantes que pueden tomar clase en el salón.
- Tipo de salón: Un salón puede ser de informática, normal, un laboratorio de física o un laboratorio de electrónica.
- Recursos de las instalaciones: Son particularidades de los salones. En Sistemas, algunos salones cuentan con televisor, tablero grande y/o persianas, inicialmente se trabaja con solo estos tres recursos, pero se deja la posibilidad de agregarlos y quitarlos.

Variables relacionadas con la asignatura:

- Intensidad horaria: Cada asignatura se dicta cierto número de horas a la semana.
- Tipos de salón requeridos por la asignatura: Existen asignaturas que requieren 2 tipos de salón diferentes, por ejemplo, el área de programación requiere de un laboratorio y un aula normal.
- Cantidad de horas que se dictarán en cada salón: Las asignaturas que necesitan diferentes tipos de salón, deben especificar cuantas horas requieren en cada uno de ellos, por ejemplo, en programación se dictan 4 horas en un salón de informática y 2 en uno normal.
- Asignaturas fraccionables: El proyecto curricular permite que ciertas asignaturas sean dictadas semanalmente por 2 docentes.

Variables relacionadas con el grupo:

- Capacidad de grupo: Cantidad máxima de estudiantes que podrán inscribirse en determinado grupo. Una asignatura está conformada por uno o varios grupos y cada uno de ellos puede diferir en la programación de horario, docente, salón y/o capacidad de estudiantes. Aunque muchas veces la capacidad del grupo depende de si la asignatura es teórico-práctica, existen ocasiones en que dicha capacidad difiere entre grupos de la misma asignatura, por ende, esta característica pertenece a cada grupo y no a la asignatura como tal.
- Asignaciones en jornadas diferentes: Se busca generar un horario compacto en el que los grupos cuya asignatura pertenece a un mismo semestre sean programados dentro de una misma jornada. Debido a que las asignaturas poseen uno o varios grupos, se buscará crear un horario donde los primeros grupos de las asignaturas de un semestre estén programados por la mañana, los segundos por la tarde, los terceros en la mañana y así sucesivamente, dependiendo de la cantidad de grupos.
- Asignaciones en días consecutivos: En el horario generado se buscará dejar al menos un día intermedio entre las asignaciones de un mismo grupo.
- Asignaciones sin espacios libres entre clases: Con el propósito de mantener un horario compacto dentro de una misma jornada, se pretende construir un horario similar al de los colegios, una vez que termina una clase continúa la otra.

Variables relacionadas con el docente:

- Cantidad de horas a asignar o Carga académica especificada: Según el tipo de vinculación, se establece cierto número de horas de clase que el docente debe dictar.
- Disponibilidad horaria del docente[3]: Franjas horarias en las que el docente está dispuesto a dictar clase.
- Asignaturas que está dispuesto a dictar según su área conocimiento.
- Recursos necesarios para la asignatura: Según criterio y enfoque pedagógico del docente, este puede requerir de los recursos de las instalaciones.

4.1 FORMULACIÓN DE LA FUNCIÓN OBJETIVO

Con base en las variables mencionadas, se plantean las restricciones que permitirán hacer el cálculo de la función objetivo y determinar la calidad de una solución. Para realizar este cálculo, en su forma más general, se busca minimizar la cantidad de

[3] Es necesario que el número de horas en la disponibilidad horaria del docente, no sea menor a la cantidad de horas a asignar. Si estas cifras son iguales, se limita el proceso y se genera una baja probabilidad de realizar la asignación de la carga académica.

restricciones duras y blandas de la solución planteada (Phuc, Khang, & Nuong, 2011) . Tal como se muestra en la Ecuación 5.

$$F.O. = MIN(RD + RB)$$

Ecuación 5: *Función Objetivo.* (Phuc, Khang, & Nuong, 2011)

En donde RD hace referencia a las restricciones duras, y RB a las restricciones blandas violadas en la solución propuesta, calculadas a través de alguna de las siguientes formas:

A nivel global o de toda la solución: Con base en todos los espacios académicos generados.

A nivel de segmento: Teniendo en cuenta únicamente los espacios que pertenecen a un mismo segmento.

A nivel individual: Revisando un único espacio académico.

Teniendo en cuenta la importancia que tienen cada una de las restricciones para la universidad, se presenta la Tabla 1 con los pesos asociados y la forma en que se realiza el cálculo en cada caso.

SIGLA	DESCRIPCIÓN	TIPO	CALCULO	PESO (W)
Dco	Docente con más de una asignación para una misma franja	DURA	SOLUCION	10
Dsc	Docente con sobre carga académica	DURA	SOLUCION	8
Dfd	Docente asignados en una franja no disponible	DURA	INDIVIDUAL	10
Dfp	Docente con asignaciones fuera de su perfil	DURA	INDIVIDUAL	10
Dsa	Docente sin alguna asignación	DURA	SOLUCION	10
Sco	Salón con más de una asignación para una misma franja	DURA	SOLUCION	10
Sfd	Salón asignado en una franja no disponible	DURA	INDIVIDUAL	10
Gsf	Grupo sin programar o espacio al que no se les asignó franja horaria	DURA	INDIVIDUAL	8
Ghi	Grupo que se ha programado sin cumplir todas las horas	DURA	SEGMENTO	10
Gti	Tipo de salón que se ha asignado de forma incorrecta	DURA	INDIVIDUAL	10
Gnf	Grupos a los que se les asignó más de un docente y no pertenecen a una asignatura fraccionable	DURA	SEGMENTO	10
ssc	Salón con sobrecupo	BLANDA	INDIVIDUAL	5
sri	Salón que no poseen los recursos pedagógicos requeridos por un docente	BLANDA	INDIVIDUAL	5
gsd	Grupos a los que no se les asignó docente	BLANDA	INDIVIDUAL	5
gjc	Asignación realizada en jornadas diferentes	BLANDA	SEGMENTO	2
gdc	Asignación realizada en días consecutivos para un mismo grupo	BLANDA	SEGMENTO	2
gel	Espacios libres o huecos entre las asignaciones de un mismo día	BLANDA	SEGMENTO	5
gss	Grupo al que no se le asignó salón	BLANDA	INDIVIDUAL	5
dci	Docente cuya carga no se asignó completamente	BLANDA	SOLUCION	3

Tabla 1*: Detalles sobre las restricciones*

Con base en las siglas presentadas y la notación descrita a continuación, se describirá el cálculo matemático de cada una las restricciones.

D:	Docente
S:	Salón
G:	Grupo
F:	Franja
E:	Espacio Académico
Z:	Segmento
R:	Recurso académico. (Puede tomar los valores D, S, G o F)
R*:	Recurso académico (Puede tomar los valores D o S)
$E_{(R)}$:	Espacio Académico en el que está asignado el recurso R
$R_{(E)}$:	Recurso académico asignado en E
W_j:	Peso de la restricción j según la Tabla 1
L_R:	Listado de R
N_R:	Número total de R, $\|L_R\| = N_R$
L_E:	Listado total de Espacios académicos
N_E:	Número total Espacios académicos, $\|L_E\| = N_E$
$L_{E(R)}$:	Listado de Espacios académicos donde está asignado R
$N_{E(R)}$:	Número de Espacios académicos donde está asignado R, $\left\|L_{E(R)}\right\| = N_{E(R)}$
L_Z:	Listado total de Segmentos
N_Z:	Número total de Segmentos, $\|L_Z\| = N_Z$
$L_{G(Z)}$:	Listado de Grupos que pertenecen al segmento Z
N_Z:	Número total de Segmentos, $\|L_Z\| = N_Z$
A_G:	Asignatura a la que pertenece el grupo G
$I_{A(G)}$:	Intensidad horaria de la asignatura a la que pertenece el grupo G
H_D:	Número de horas que se deben asignar a D
T_{R^*}:	Conjunto de franjas disponibles o asignables del recurso R* (disponibilidad de R*)
P_D:	Lista de asignaturas que puede dictar el docente D (perfil de D)
Θ:	Tipo de Salón
$\Omega_{(G)}$:	Tipos de salón asignados a G
$H_\Theta A_G$:	Horas a asignar del tipo de salón C según la asignatura a la que pertenece el grupo G
$N_{E(\Theta, G)}$:	Numero de Espacios Académicos asignados al grupo G cuyo salón es del tipo Θ
$L_\Theta A_G$:	Listado de tipos de salón requeridos por la asignatura a la que pertenece el grupo G
$N_{E(\Theta, G)}$:	Numero de Espacios Académicos asignados
Δ_S:	Cantidad de estudiantes que puede albergar S (capacidad de S)
Δ_G:	Número de estudiantes que conforman G
Γ_S:	Conjunto de recursos pedagógicos con los que cuenta S
$\Gamma_D A_G$:	Conjunto de recursos pedagógicos requerido por el docente D para dictar la asignatura a la que pertenece el grupo G
$\Pi_{(F)}$:	Hora de inicio de F
$\Phi_{(F)}$:	Hora Final de F
Ψ_F:	Número de Día de la semana de F
λ_Z :	Número o posición que ocupa el segmento Z en la lista total de segmentos
L_Θ:	Listado de todos los tipos de salón
L_Ψ:	Listado de días de la semana
$L_{E(\Psi, Z)}$:	Listado de Espacios del segmento Z programados en el día Ψ
$L_{D(G)}$:	Listado de docentes asignados al gupo G

***Ilustración 5**: Notación Utilizada para la representación del cálculo de la función objetivo*

El número de Espacios Académicos está determinado por el producto entre la cantidad de grupos existentes y la intensidad horaria de la asignatura a la que pertenece el grupo. Así, un grupo con una intensidad horaria de 6 horas semanales requerirá de la creación de 3 espacios académicos; Teniendo en cuenta que en esta implementación las franjas manejadas son de 2 horas y que en caso de requerir un número de horas impar, se redondeará al número inmediatamente superior. (Si se requieren 3 horas se deben asignar 2 espacios académicos, cada uno con una franja de 2 horas sumando en total 4 horas).

$$N_E = \frac{\sum_{G_i \in L_G} I_{A(G_i)}}{2}$$

***Ecuación 6**: Número total de espacios que se deben crear*

Para calcular los docentes o salones que poseen más de una asignación en una misma franja (colisión), se define la siguiente ecuación:

$$Col_{R*}(E_i) = \begin{cases} 1, & Si\ \forall E_{i,j} \in L_{E(R*)}, \quad \exists E_j \mid F_{(E_i)} = F_{(E_j)} \\ 0, & En\ caso\ contrario \end{cases}$$

***Ecuación 7**: Cálculo de colisiones de un docente o salón*

4.1.1 RESTRICCIONES DURAS (RD)

En la Ecuación 8 se desglosa una parte de la Ecuación 5 de tal manera que se representan las restricciones duras como la sumatoria de aquellas relacionadas con los docentes, salones y asignaturas.

$$RD = rdd + rds + rdg$$

***Ecuación 8:** Ecuación de cálculo de restricciones duras*

Dónde:

rdd = Restricciones duras por docente
rds = Restricciones duras por salón
rga = Restricciones duras por grupo

Teniendo en cuenta que una solución solo será factible si la sumatoria de las restricciones duras es igual a 0.

$$RD = 0$$

Ecuación 9: *Factibilidad de una solución*

4.1.1.1 *Restricciones Duras por Docente*

La Ecuación 10 presenta el cálculo de las restricciones duras por docente, teniendo en cuenta las asignaciones con sobre carga académica, en franjas no disponibles, fuera del perfil del docente o que presentan colisiones (varias asignaciones en una misma franja) y aquellos docentes a los que no se les realizó ninguna asignación.

$$rdd = \sum_{D_i \in L_D} Dco_{D_i} * \ W_{Dco} + \sum_{D_i \in L_D} Dsc_{D_i} * \ W_{Dsc} + \sum_{E_i \in L_E} Dfd_{E_i} * \ W_{Dfd} + \sum_{E_i \in L_E} Dfp_{E_i} * \ W_{Dfp} + \sum_{D_i \in L_D} Dsa_{D_i} * \ W_{Dsa}$$

Ecuación 10: *Cálculo de restricciones duras por docente*

Dónde:

Dco (docentes con colisiones): Un docente no puede tener asignados 2 grupos en la misma franja horaria. Se calcula con base en la Ecuación 7:

$$Dco_D = \sum_{E_i \in L_{E(D)}} Col_D(E_i)$$

Ecuación 11*: Calculo de la restricción dura Dco, docentes con colisiones*

***Dsc (docentes con sobrecarga académica)*:** Las horas asignadas a un docente no deben exceder la carga académica especificada.

$$Dsc_D = \begin{cases} 1, & Si\ H_D < N_{E(D)} \\ 0, & En\ caso\ contrario \end{cases}$$

Ecuación 12*: Calculo de la restricción dura Dsc, docentes con sobrecarga académica*

Dfd (docentes con asignaciones fuera de su disponibilidad): Un docente solo puede ser programado en las franjas horarias en que está disponible.

$$Dfd_E = \begin{cases} 1, & Si\ F_{(E)} \notin T_{D(E)} \\ 0, & En\ caso\ contrario \end{cases}$$

Ecuación 13: *Cálculo de la restricción dura Dfd, docentes con asignaciones fuera de su disponibilidad*

Dfp (docentes con asignaciones fuera de su perfil): Un docente solo puede dictar clases que se acomoden a su perfil (área de conocimiento).

$$Dfp_E = \begin{cases} 1, & Si\ A_{G(E)} \notin P_{D(E)} \\ 0, & En\ caso\ contrario \end{cases}$$

Ecuación 14: *Cálculo de la restricción Dfp, docentes con asignaciones fuera de su perfil*

Dsa (docentes sin asignar): Todos los docentes deben ser asignados.

$$Dsa_D = \begin{cases} 1, & Si\ N_{E(D)} = 0 \\ 0, & En\ caso\ contrario \end{cases}$$

Ecuación 15: *Cálculo de la restricción Dsa, docentes sin asignar*

4.1.1.2 *Restricciones Duras por Salón*

En la Ecuación 8 se puede observar la fórmula que representa las restricciones duras relacionadas con los salones de clase, en la cual se calculan aquellos que tienen asignado más de un espacio académico en la misma franja horaria y los que se programan en una hora en la que el salón no se encuentra disponible.

$$rds = \sum_{S_i \in L_S} Sco_{S_i} * \ W_{Sco} + \sum_{E_i \in L_E} Sfd_{E_i} * \ W_{Sfd}$$

Ecuación 16: *Cálculo de restricciones duras por salón*

Siendo:

***Sco (salones con colisiones)**:* Un salón no debe tener 2 asignaciones en la misma franja horaria. Se calcula con base en la Ecuación 7:

$$Sco_S = \sum_{E_i \in L_{E(S)}} Col_S(E_i)$$

***Ecuación 17:** Cálculo de la restricción dura Sco, salones con colisión*

Sfd (salones con asignaciones fuera de su disponibilidad): Un salón solo puede ser programado en las franjas horarias en que está disponible.

$$Sfd_E = \begin{cases} 1, & Si \ F_{(E)} \notin T_{S(E)} \\ 0, & En\ caso\ contrario \end{cases}$$

***Ecuación 18:** Cálculo de la restricción Sfd, salones con asignaciones fuera de su disponibilidad*

4.1.1.3 *Restricciones Duras por Grupo*

Para el caso de las restricciones duras por grupo, en la Ecuación 9 se presenta la sumatoria de grupos que no han sido programados por el algoritmo, el número de grupos que han sido programados de manera incompleta o sin cumplir con la intensidad horaria de la asignatura a la que pertenecen, los grupos a los cuales se les ha asignado un tipo de salón de manera incorrecta y aquellos que no son fraccionables y se les asignaron 2 docentes.

$$rdg = \sum_{E_i \in L_E} Gsf_{E_i} * \ W_{gsf} + \sum_{Z_i \in L_Z} Ghi_{Z_i} * \ W_{Ghi} + \sum_{z_i \in L_Z} Gti_{Z_i} * \ W_{Gti} + \sum_{z_i \in L_Z} Gnf_{Z_i} * \ W_{Gnf}$$

***Ecuación 19:** Calculo de restricciones duras por asignatura*

Dónde:

Gsf (grupos sin franja): Grupos sin programar o espacios a los que no se les asignó una franja.

$$Gsf_E = \begin{cases} 1, & Si\ \ F_{(E)}\ no\ existe\ o\ es\ nula \\ 0, & En\ caso\ contrario \end{cases}$$

***Ecuación 20**: Cálculo de la restricción Gsf, grupos sin franja*

Ghi (Grupos con horas incompletas): Para cada grupo se deben asignar exactamente la cantidad de horas especificadas en la intensidad horaria de la asignatura a la que pertenece. Un grupo con horas incompletas es aquel para el que no se creó la totalidad de Espacios Académicos requeridos.

$$GHi_Z = \sum_{G_i \in L_{G(Z)}} I_{A(G)} - N_{E(G)}\,, \quad I_{A(G)} \geq N_{E(G)}$$

***Ecuación 21**: Cálculo de la restricción Ghi, grupos con horas incompletas*

Gti (Grupos a los que se les asignó un tipo de salón incorrecto): Al momento de crear una asignatura, se deben indicar los tipos de salón y la cantidad de horas requeridas para cada uno de ellos, teniendo en cuenta que la sumatoria de las horas requeridas por los diferentes tipos de salón debe ser igual a la intensidad horaria de la asignatura.

$$\sum_{\Theta_i \in L_\Theta A_G} \mathrm{H}_{\Theta i}\ \mathrm{A}_\mathrm{G} = I_{A(G)}$$

***Ecuación 22**: Relación entre los tipos de salón requeridos y la intensidad horaria de una asignatura*

Esto quiere decir que la cantidad de horas requeridas por un tipo de salón debe ser menor o igual a la intensidad horaria de la asignatura.

$$\forall \Theta \in \mathrm{L}_\Theta, \qquad \mathrm{H}_\Theta\ \mathrm{A}_\mathrm{G} \leq I_{A(G)}$$

***Ecuación 23**: Restricción de las horas requeridas por un tipo de salón*

De este modo, un grupo con tipo de salón incorrecto es aquel al que no se les asignó el tipo de salón requerido por la asignatura. Si una asignatura requiere de 2 horas de un salón normal y 4 horas de laboratorio, y a los tres espacios requeridos (ver Ecuación 6) se les asigna un salón normal, se presentan 2 violaciones a la restricción ya que 2 de los espacios deberían tener asignado un salón de tipo laboratorio.

$$Gti_Z = \sum_{G_i \in L_{G(Z)}} tiz(G_i)$$

Ecuación 24: *Cálculo de la restricción Gti, grupos a los que se les asignó un tipo de salón incorrecto*

La Ecuación 24 describe el cálculo de esta restricción teniendo en cuenta las dos ecuaciones siguientes:

$$tiz(G) = \sum_{\Theta_i \in \Omega_{(G)}} Tig(\Theta_i, G)$$

Ecuación 25: *Cálculo de los tipos de salón incorrectos asignados a un grupo*

$$TiG(\Theta, G) = \begin{cases} n, & Si\ n > 0, \quad n = \frac{\mathrm{H}_{\Theta}\ \mathrm{A}_{\mathrm{G}}}{2} - N_{E(\Theta,G)} \\ 0, & En\ caso\ contrario \end{cases}$$

Ecuación 26: *Violación de las horas que requiere un grupo de cierto tipo de salón*

La primera ecuación describe el cálculo de las violaciones por grupo mientras que la segunda hace referencia al cálculo por el tipo de salón. Básicamente por cada tipo de salón asignado a un grupo (Ecuación 25), se calcula la violación de las horas que se requieren de dicho tipo de salón (Ecuación 26).

La Ecuación 26 estima la diferencia entre la cantidad de Espacios académicos que deberían tener asignado el tipo de salón, y el número de espacios que poseen dicha asignación. Para entender por qué esta diferencia solo se contabiliza al ser mayor que 0, se retoma el ejemplo en el que una asignatura requiere 2 horas de salón normal y 4 de laboratorio, pero todos los salones asignados corresponden al primer tipo:

Θ_1 = Salón normal

Θ_2 = Laboratorio

$$\frac{\mathrm{H}_{\Theta 1}\,\mathrm{A}_{\mathrm{G}}}{2} - N_{E(\Theta 1,G)} = \frac{2}{2} - 3 = -2$$

$$\frac{\mathrm{H}_{\Theta 2}\,\mathrm{A}_{\mathrm{G}}}{2} - N_{E(\Theta 2,G)} = \frac{4}{2} - 0 = 2$$

***Ilustración 6**: Ejemplo sobre el cálculo de las violaciones en las horas que requiere un grupo de cierto tipo de salón*

Al realizar el cálculo de las violaciones con respecto a un laboratorio se obtiene como resultado 2, lo que corresponde con el número de tipos de salón que no se asignaron. El valor -2 obtenido al hacer el cálculo del salón normal también representa las 2 violaciones, sin embargo no se podría contabilizar porque sería redundante. Por tal razón solo se tienen en cuenta los valores positivos.

Gnf (Grupos no fraccionables a los que se les asignaron dos docentes): Solo los grupos que pertenecen a una asignatura fraccionable pueden tener asignados 2 docentes.

$$Gnfi_Z = \sum_{G_i \in L_{G(Z)}} nfr(G_i)$$

***Ecuación 27**: Cálculo de la restricción Gnf, grupos no fraccionables a los que se les asignaron dos docentes*

$$nfr(G) = \begin{cases} 1, & Si\ \left|L_{G(D)}\right| = 2\ \wedge\ A_G\ no\ es\ fraccionable \\ 0, & En\ caso\ contrario \end{cases}$$

***Ecuación 28**: Calculo de un grupo no fraccionable al que se le asignaron 2 docentes*

La Ecuación 27 representa el cálculo para todos los grupos que conforman un segmento, mientras que la Ecuación 28 determina el cálculo para un solo grupo; si este no es fraccionable y tiene asignados 2 docentes se considera como una violación a la restricción mencionada

4.1.2 RESTRICCIONES BLANDAS (RB)

La ecuación 10 representa las restricciones blandas como la suma de aquellas que están relacionadas con los salones, las restricciones concernientes a las asignaturas y las violaciones respecto a los docentes.

$$RB = rbs + rbg \ + rbd$$

Ecuación 29: *Cálculo de restricciones blandas*

Dónde:

rbs = restricciones blandas por salón
rbg = restricciones blandas por grupo
rbd = restricciones blandas por docente

4.1.2.1 *Restricciones Blandas por Salón*

En la Ecuación 30 se puede observar en detalle la representación matemática de las restricciones blandas que se relacionan con los salones, en donde intervienen la cantidad de salones que tienen sobrecupo de estudiantes y la cantidad de salones que no cuentan con los recursos académicos solicitados por la asignatura y el docente.

$$rbs = \sum_{E_i \in L_E} ssc_{E_i} * \ W_{ssc} + \sum_{E_i \in L_E} sri_{E_i} * \ W_{sri}$$

Ecuación 30: *Cálculo de restricciones blandas por salón*

Siendo:

ssc (Salones con sobrecupo): La capacidad del salón asignado debe ser preferiblemente igual o mayor al número de estudiantes que conforman el grupo. En los horarios implementados por la universidad se han visto casos en los que se observa hacinamiento de los estudiantes debido a que se asigna un salón muy pequeño. Por esta razón se considera una restricción blanda.

$$ssc_E = \begin{cases} 1, & Si \ \Delta_{S(E)} < \Delta_{G(E)} \\ 0, & En\ caso\ contrario \end{cases}$$

Ecuación 31*: Cálculo de la restricción ssc, salones con sobrecupo*

sri (Salones con recursos pedagógicos incompletos): Los salones asignados deben cumplir con los recursos requeridos por el enfoque pedagógico del docente.

$$sri_E = \left|\Gamma_{S(E)} \setminus \Gamma_{D(E)} A_{G(E)}\right|$$

***Ecuación 32**: Cálculo de la restricción sri, salones con recursos pedagógicos incompletos*

4.1.2.2 *Restricciones Blandas por Grupo*

La Ecuación 33 muestra las variables que intervienen en la representación de las restricciones blandas por asignatura, en la cual se realiza una sumatoria de los grupos que no cuentan con un docente, las asignaturas que son dictadas en días consecutivos de la semana, los espacios libres o huecos presentes en la asignación de los espacios académicos y las asignaciones programadas en jornadas diferentes (mañana o tarde).

$$rbg = \sum_{E_i \in L_E} gsd_{E_i} * \ W_{gsd} + \sum_{Z_i \in L_Z} gjc_{Z_i} * \ W_{gjc} + \sum_{Z_i \in L_Z} gdc_{Z_i} * \ W_{gdc} + \sum_{Z_i \in L_Z} gel_{Z_i} * \ W_{gel} + \sum_{E_i \in L_E} gss_{E_i} * \ W_{gss}$$

***Ecuación 33:** Cálculo de restricciones blandas por asignatura*

Dónde:

***gsd (grupos sin docente asignado)*:** Un grupo debe tener asignado por lo menos un docente. En algunas ocasiones la Universidad abre concursos para seleccionar un docente después de que se ha llevado a cabo el proceso de inscripción de los estudiantes, por ende es posible que aunque se precisa el grupo, el salón y el horario, no se tenga un docente asignado

$$gsd_E = \begin{cases} 1, & Si \ D_{(E)} \ no \ existe \ o \ es \ nulo \\ 0, & En \ caso \ contrario \end{cases}$$

***Ecuación 34:** Cálculo de la restricción gsd, grupos sin docente asignado*

gjc (grupos con asignaciones en jornada contraria): Debido a que algunos estudiantes deben trabajar, resulta útil que puedan ver todas las asignaturas de su semestre en una

misma jornada. Dicho de otra manera, los grupos que pertenecen a asignaturas de un mismo semestre deben programarse en una misma jornada (mañana o tarde).

$$gjc_Z = \sum_{E_i \in L_{E(Z)}} Jc(E_i, Z)$$

***Ecuación 35**: Cálculo de la restricción gjc, grupos con asignaciones en jornada contraria*

$$Jc(E,Z) = \begin{cases} 1, Si\ \left(\lambda_{(Z)}\ mod\ 2 = 0\ \wedge\ \Pi_E > 12\ pm\right) \vee\ \left(\lambda_{(Z)}\ mod\ 2 \neq 0\ \wedge\ \Pi_E < 12\ pm\right) \\ 0, \qquad\qquad En\ caso\ contrario \end{cases}$$

***Ecuación 36**: Calculo de la violación de un espacio académico asignado en una jornada contraria*

Para el cálculo de esta restricción, a cada segmento se le asigna un número que coincide con la posición en la lista de todos los segmentos creados; los grupos que pertenecen a segmentos con posiciones pares deben ser programados en la jornada de la mañana, mientras que los ubicados en posiciones impares deben estar asignados en la jornada de la tarde (Ecuación 36). Con esta implementación los segmentos de un mismo semestre pueden estar conformados por un número diferente de grupos; si cada asignatura del semestre x posee 2 grupos, pero alguna de ellas tiene un tercero, se crearán 3 segmentos diferentes, donde, los dos primeros estarán conformados con el mismo número de grupos mientras el último poseerá un solo elemento.

Segmento 1 = {grupo 1 Cálculo **1**, grupo 1 Física **1**,…., Grupo 1 Catedra **1**}
Segmento 2 = {grupo 2 Cálculo **1**, grupo 2 Física **1**,…., Grupo 2 Catedra **1**}
Segmento 3 = {grupo 3 Cálculo **1**}

Aunque se presenta esta situación, con los dos primeros segmentos se garantiza la posibilidad de ver todas las asignaturas de un mismo semestre, el primero y el tercero estarán programados en la jornada de la mañana, mientras que el segundo estará en la jornada de la tarde. Un estudiante inscrito en el grupo 3 de Cálculo 1, tendrá la garantía de ver todas las asignaturas si se inscribe al resto de grupos de los segmentos 1 o 2. No obstante, al tomar los grupos del segundo segmento perdería la posibilidad de estudiar en una sola jornada. Si alguna asignatura posee más grupos que el resto de las asignaturas del mismo semestre, es probable que se presenten violaciones a esta restricción. Por tal razón se considera como blanda.

gdc (grupos con asignaciones en días consecutivos): Un grupo puede poseer varios Espacios Académicos debido a la cantidad de horas que deben ser programadas. Si para el grupo se requieren 2 horas, solo necesitará un espacio, si requiere de 4 horas, 2 espacios y así sucesivamente. Si entre las demás asignaciones de un mismo grupo no existe al menos un día intermedio entre las franjas programadas, se dice que existe una asignación en día consecutivo.

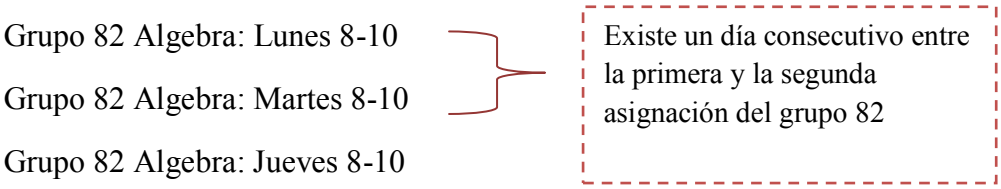


Ilustración 7: Ejemplo de restricción "Día consecutivo"

Para el cálculo de esta restricción se definen las Ecuación 37, Ecuación 38 y Ecuación 39:

$$gdc_Z = \sum_{G_i \in L_{G(Z)}} diz(G_i)$$

Ecuación 37: *Cálculo de la restricción gdc, grupos con asignaciones en días consecutivos*

$$diz(G) = \sum_{i=0}^{|L_{E(G)}|-1} dc_G(E_i, E_{i+1}) \ , \qquad E_{i,i+1} \in L_{E(G)}$$

Ecuación 38: *Calculo de días consecutivos de un grupo*

$$dc_G(E_i, E_j) = \begin{cases} 1, & |\psi_{F(Ei)} - \psi_{F(Ej)}| < 2 \\ 0, & En\ caso\ contrario \end{cases} , \qquad E_{i,j} \in L_{E(G)}$$

Ecuación 39: *Cálculo de días consecutivos entre 2 Espacios Académicos de un mismo grupo*

La primera ecuación calcula el número de violaciones en un segmento, por lo que hace uso de la Ecuación 38 que determina el número de violaciones por grupo y que depende a su vez de la Ecuación 39, donde se obtiene la violación entre 2 espacios pertenecientes al mismo grupo.

***gel (grupos con espacios libres)*:** Mide la cantidad de horas existentes entre 2 espacios que pertenecen a un mismo segmento. Si por ejemplo, para el día lunes el primer espacio es de 6 a 8 y el segundo de 12 a 14, se tendrá un valor de 4 (12 – 8) horas libres en las que el estudiante no tomará clases.

$$gel_Z = \sum_{\Psi_i \in L_\Psi} el(\Psi_i, Z)$$

Ecuación 40: *Cálculo de la restricción gel, grupos con espacios libres*

Para obtener el número de horas libres entre 2 espacios, basta con restarle a la máxima hora de inicio la mínima hora final, tal y como se observa en la siguiente ecuación.

$$el(\Psi, Z) = \sum_{i=0}^{|L_{E(\Psi,Z)}|-1} Max(\Pi_{Ei}, \Pi_{Ei+1}) - Min(\Phi_{Ei}, \Phi_{Ei+1}), \quad E_{i,i+1} \in L_{E(z)}$$

Ecuación 41*: Número de horas libres entre 2 espacios pertenecientes al mismo segmento*

***gss (grupos sin un salón asignado)*:** Durante la inscripción de asignaturas por parte de los estudiantes, se presentan varios casos en los que aún no se tienen laboratorios asignados, y debido a que el problema tratado en este documento tiene lugar antes de dicha inscripción, este apartado se toma como una restricción blanda.

$$gss_E = \begin{cases} 1, & Si\ \ S_{(E)}\ no\ existe\ o\ es\ nulo \\ 0, & En\ caso\ contrario \end{cases}$$

Ecuación 42*: Cálculo de la restricción gss, grupos sin un salón asignado*

4.1.2.3 *Restricciones Blandas por docente*

La ecuación 13 presenta el cálculo de las restricciones blandas por docente.

$$rbd = \sum_{D_i \in L_D} dci_{D_i} * \ W_{dci}$$

Ecuación 43: *Cálculo de restricciones blandas por docente*

Siendo:

dci (docentes con cargas incompletas): Al momento de registrar o crear un docente en la aplicación, el usuario debe incluir el número de horas que deben ser asignadas. Un docente con carga incompleta será aquel a quien no se le asignó la totalidad de horas especificadas.

$$dci_D = \begin{cases} H_D - N_{E(D)}, & Si\ H_D > N_{E(D)} \wedge N_{E(D)} \neq 0 \\ 0, & En\ caso\ contrario \end{cases}$$

Ecuación 44: *Cálculo de la restricción dci, docentes con cargas incompletas*

En la información entregada por la universidad se observaron tres casos en los que un docente no tenía la totalidad de la carga asignada. Así mismo, para poder cumplir con esta restricción, la totalidad de horas a asignar(número de grupos) debe ser igual a la carga académica total de los docentes, si hay más carga que grupos se tendrán cargas incompletas, pero si el número de grupos es mayor se tendrán asignaciones sin docente. Por estas dos razones se toma esta restricción como blanda.

4.2 IMPLEMENTACIÓN DE LAS TÉCNICAS

El objetivo de cada algoritmo será construir y/o modificar un conjunto de Espacios Académicos para satisfacer el mayor número de restricciones posibles y de esa forma obtener el menor valor de la función objetivo.

4.2.1 ALGORITMO VORAZ

El algoritmo Voraz o Greedy, es una técnica programada para generar una solución en un tiempo de ejecución pequeño con respecto a Búsqueda Tabú, Algoritmos Genéticos y Colonia de Hormigas. En el presente desarrollo se usa como primer acercamiento a un resultado, siendo el sustento para generar nuevas soluciones.

Se optó por una metodología propia de asignación basada en fases, en las cuales se van asignando recursos académicos del mismo tipo a todos los espacios académicos establecidos. En la primera fase se asignaron docentes a los grupos, en la segunda franjas horarias a la anterior dupla y por último se realizó una asignación de salones a todos los Espacios Académicos.

4.2.1.1 *Asignación Por Fases*

Se planteó una asignación por fases, debido a que hay mayor control sobre las posibles soluciones arrojadas por el algoritmo, por ejemplo, en la primera fase, el número de docentes para el caso de prueba del desarrollo es 80, así, para asignar un docente a un espacio académico, el algoritmo cuenta a lo sumo con 80 posibles asignaciones por cada uno de los espacios existentes.

4.2.1.1.1 Fase I

En esta fase el algoritmo asigna docentes a los grupos de cada asignatura teniendo en cuenta el tipo de vinculación y las restricciones planteadas (véase sección "Modelamiento de la solución").

El proceso se inicia organizando la lista de docentes que se obtiene de la base de datos, dándole prioridad a aquellos cuya dedicación sea de tiempo completo.

Después de tener la lista de docentes organizada, el algoritmo selecciona un grupo de forma aleatoria y asigna el primer docente que pueda dictar la asignatura a la que este pertenece. El proceso se repite hasta llegar a alguna de las condiciones de parada (que no haya grupos, que los docentes queden con su carga académica completa o que no existan candidatos con disponibilidad). Para los grupos de las asignaturas fraccionables se buscan dos docentes que tengan disponibilidad y que estén en capacidad de dictarlos.

4.2.1.1.2 Fase II

Después que la primera fase del algoritmo Greedy (asignación de docentes a los diferentes grupos de cada asignatura) arroje una solución y teniendo en cuenta las restricciones planteadas, la segunda fase se encarga de asignar una franja horaria a cada uno de los espacios académicos.

La asignación de estas franjas se realiza teniendo en cuenta la disponibilidad de los docentes asignados a cada dupla (docente y grupo). En caso que el espacio académico no tenga un docente asignado, el algoritmo toma como referencia todas las franjas que fueron planteadas en la configuración general del aplicativo[4].

El proceso inicia con la organización de los espacios académicos entregados por la primera fase del algoritmo Greedy. Esta organización se realiza teniendo en cuenta el semestre de la asignatura a la cual pertenece el grupo, de tal manera que los espacios académicos estén organizados ascendentemente según el semestre al que pertenece la asignatura.

[4] Para más Información sobre la configuración general del aplicativo creado en este proyecto, ir a la sección de anexos, apartado: Ingreso de Datos al Sistema.

Después de tener los datos previamente organizados se realiza la asignación utilizando un proceso iterativo, que consiste en asignar franjas, de preferencia por la mañana, a todos los primeros grupos de las asignaturas en cada uno de los semestres. Después de realizar todas las asignaciones de franjas hasta las asignaturas del décimo semestre se asignan todos los segundos grupos de las asignaturas en cada semestre (si existe más de un grupo por asignatura) pero esta vez con preferencia en el horario de la tarde. Los terceros grupos tendrán preferencia por la mañana, los cuartos en la tarde y así sucesivamente según los grupos existentes en cada asignatura.

La razón por la cual se optó por utilizar esta metodología en la asignación de franjas, radica en la mayor probabilidad de satisfacer la restricción blanda en la que las asignaturas de un mismo semestre deben programarse en igual jornada y de forma consecutiva, ya sea en la mañana o en la tarde. Adicionalmente el algoritmo realiza un control de las asignaciones en el que tiene en cuenta los grupos pertenecientes cada segmento. De tal manera que se garantiza en cada iteración que las asignaciones de franjas a grupos de un semestre equivalente no colisionen. Esto con el fin de garantizar que un estudiante pueda cursar todas las asignaturas correspondientes a cada semestre sin que se presenten cruces de horario en las materias.

4.2.1.1.3 Fase III

En esta fase el algoritmo hace la asignación del salón al espacio académico (docente, grupo y hora), teniendo en cuenta la disponibilidad del salón.

En primer lugar se ordenan los salones ascendentemente respecto a la cantidad de horas disponibles y luego se agrupan por tipo de salón, para asignarlos a los grupos que necesiten más de un tipo (grupos que necesiten aulas de laboratorios). Durante este proceso se tiene en cuenta que el salón esté disponible en la franja asignada, que su capacidad sea mayor al número de estudiantes que conforman el grupo y que posea los recursos requeridos por el docente establecido previamente.

4.2.1.2 *Características de la Asignación*

La gran mayoría de los elementos que componen la asignación del algoritmo están relacionados con las características de la metodología general de Greedy, sin embargo es necesario resaltar la organización de las listas de los recursos que conforman un espacio académico (Docentes, Franjas, Salones y Grupos). Esta organización se hace de tal manera que la asignación de recursos se realice en un orden específico, favoreciendo las probabilidades de llegar a una buena solución. Para el caso de la primera fase, se organizó la lista de grupos dando prioridad a aquellas asignaturas con menor número de docentes candidatos. Después de tener esta dupla asignada, se organizaron los espacios académicos por semestre y asignatura, para que el algoritmo estableciera las franjas

horarias, de tal manera que se intercalaran espacios académicos en horas de la mañana y en horas de la tarde. Finalmente los espacios académicos fueron organizados dando preferencia a aquellos que necesitaban de un salón con características especiales (alta capacidad, varios recursos y/o laboratorios).

Para el caso de los docentes se dio prioridad a aquellos que son de planta, por lo que se asignan en primera instancia los que tienen este tipo de vinculación con la institución educativa.

4.2.1.3 *Estructuras Utilizadas*

Algunas de las estructuras utilizadas y que sirvieron de apoyo para realizar la correcta asignación de los espacios académicos están altamente relacionadas con el modelo de clases de la lógica del negocio (Ver Ilustración 54).

Además de las estructuras representadas en este diagrama, también se utilizaron listas, en las cuales se almacenaban todos los datos correspondientes a los docentes, salones, grupos y franjas horarias que fueron ingresados en la base de datos previamente.

Finalmente, la solución es almacenada en una lista de espacios académicos vacíos (sin datos), que van siendo complementados con todos los recursos necesarios a medida que el algoritmo escala a través de las tres fases.

4.2.1.4 *Presentación de la Solución*

Este algoritmo tiende a generar un pequeño número de asignaciones sin franja horaria y docentes con carga incompleta. Sin embargo, al ser usado como materia prima de los otros algoritmos, serán estos quienes se encarguen de garantizar el cumplimiento de dicha restricción.

4.2.2 BÚSQUEDA TABÚ

En la implementación de esta técnica se siguió el enfoque original de búsqueda tabú adaptativa (Lü & Hao, 2008) realizando algunos ajustes sobre dicho. El primero de ellos consistió en la implementación de la "lista de frecuencias", el segundo en la manipulación de la lista tabú y el tercero en el incremento de la profundidad.

Se intentó recuperar uno de los aspectos básicos del modelo general de TS denominado "lista de frecuencias" (Glover & Laguna, 1997) cuyo objetivo es brindar información sobre la cantidad de veces que una solución es visitada o que un movimiento es realizado (Glover, Laguna, & Martí, 2007) . En el algoritmo ATS original no existe como tal una lista de frecuencias, sino que la frecuencia con la que se visita una

solución influye en la duración de un elemento en la lista tabú. La lista de frecuencias implementada indicaba la cantidad de veces que se ha realizado un movimiento o asignación a un grupo en particular, sin embargo, se obtuvieron mejores resultados al eliminar dicha lista debido a que se requería de un número de iteraciones muy alto para aprovechar la información almacenada en ella. Como consecuencia, el algoritmo implementado solo contempla la lista tabú como estructura de memoria, tal y como ocurre en el ATS original (Lü & Hao, 2008).

Aunque la primera variación no hace parte del algoritmo ATS implementado, los otros dos ajustes si tuvieron lugar.

En el algoritmo original se establece una fórmula matemática que determina la duración de un elemento en la lista tabú, basado en la frecuencia con que ha sido visitado. Este enfoque fue modificado estableciendo la lista tabú como una cola (los primeros elementos en ingresar son los primeros en salir) y agregando un parámetro denominado "tamaño de lista tabú" que determina la cantidad de elementos que podrán estar en dicha cola.

La última variación al algoritmo consistió en modificar el momento en que el valor de la profundidad es incrementado. En el algoritmo original se incrementa dicho valor cuando se consigue una solución prometedora, es decir, se reduce la función objetivo de la mejor solución, sin embargo, en la implementación realizada se observaron mejores resultados al incrementar la profundidad cuando no es posible obtener un candidato para modificar la solución. En la sección 4.2.5 se abordan con más detalle los resultados de las pruebas realizadas.

4.2.2.1 *Búsqueda tabú adaptativa*

Partiendo del enfoque ATS presentado por Lü & Hao (2008) se incorporaron los criterios de adaptación denominados "profundidad" y "fuerza de perturbación" aplicados de la siguiente forma:

Profundidad (α): Indica la cantidad de candidatos que se crearán a partir un grupo seleccionado. Cada candidato estará conformado por n movimientos determinados por el número de Espacios Académicos que se deben crear para el grupo seleccionado. De esta manera, si un grupo requiere de 3 Espacios, cada candidato contará con 3 movimientos.

Al cambio realizado a un Espacio Académico se denomina "movimiento". Si para el Espacio (docente A - salón A - grupo A - franja A), se realiza un cambio por el docente B (docente **B** - salón A - grupo A - franja A) por ejemplo, se dice que se realizó un movimiento por docente. Si para ese mismo espacio se realiza un cambio por docente, franja y salón (docente **B** - salón **B** - grupo A - franja **B**) se dice que se realizó un movimiento completo. Los grupos como tal no son modificados dado que estos determinan la cantidad de espacios que deben ser creados por el algoritmo.

La profundidad define entonces la cantidad de candidatos que se generan a partir de las asignaciones de un grupo seleccionado, de modo que si la profundidad es 3, se crearán 3 candidatos. Los movimientos que conforman cada candidato son considerados "movimientos factibles" ya que deben satisfacer todas las restricciones duras.

Fuerza de perturbación (β): Indica al algoritmo el número de grupos que se tomarán para la creación de la lista de candidatos. Si por ejemplo se tiene una profundidad 3 y una fuerza de perturbación 4, se tendrán cuatro grupos diferentes en la solución, y para cada uno de ellos se buscarán tres candidatos diferentes. Esto generará una lista de candidatos conformada por 12 elementos determinados por el producto de la fuerza de perturbación y la profundidad, 4 * 3 en el ejemplo anterior.

Estas 2 variables buscan potencializar los mecanismos de intensificación y diversificación de la búsqueda tabú adaptándose de la siguiente forma:

Diversificación: Cuando el mejor candidato no consigue reducir la función objetivo de la mejor solución, se incrementa el valor de β para orientar la búsqueda de candidatos sobre un número mayor de grupos.

Intensificación: El valor de α se incrementa cuando no es posible obtener algún candidato, ya sea porque no es posible realizar nuevas asignaciones o porque los candidatos creados están en la lista tabú y no superan el criterio de aspiración. De esta manera se realizará un mayor esfuerzo para generar candidatos a partir de cada uno los grupos determinados por β.

La profundidad α y la fuerza de perturbación β estarán restringidos por sus respectivos parámetros de valor máximo, mínimo y e incremento.

4.2.2.2 *Evaluación de la Función Objetivo*

Debido al costo que requeriría evaluar la función objetivo de todo el conjunto de espacios académicos creados, en cada iteración del algoritmo, se recurrió a medir el impacto generado al incluir un candidato en la solución. Al inicio del algoritmo se calcula la función objetivo de la solución (con la totalidad de las restricciones violadas), la función objetivo de cada segmento (con la totalidad de las violaciones de un segmento) y la función objetivo de cada espacio (con violación de las restricciones individuales). Con base en ellos se mide el impacto de cada candidato de la siguiente manera:

- **Restricciones individuales:** Por cada movimiento que conforma al candidato, se calculan las restricciones Dfd, Dfp, Sfd, Gsf, Gti, ssc, sri, gsd y gss. Una vez que se tienen estos valores, a la función objetivo de cada movimiento, se le restan los valores de la función objetivo de cada Espacio Académico (del grupo seleccionado):

	Dfd	Dfp	Sfd	Gsf	ssc	sri	gsd	gss
Movimiento Creado	0	0	0	0	0	0	1	0
Espacio Asignado	1	0	0	0	1	2	1	0
	-1	0	0	0	-1	-2	0	0

***Ilustración 8**: Calculo del impacto en las restricciones individuales*

Los valores obtenidos se suman a la función objetivo del segmento al que pertenece el grupo, manteniéndolo actualizado con respecto a las restricciones individuales.

- **Restricciones por segmento:** Se reemplazan los Espacios Académicos asignados por los movimientos creados y se calculan las restricciones Ghi, Gnf, gjc, gdc, y gel. A estos valores se restan las violaciones de cada restricción en la función objetivo del segmento previamente asignado, teniendo en cuenta que las valores de las restricciones individuales han sido actualizados en el paso anterior.

	Ghi	Gnf	gjc	gdc	Restricciones Individuales
Segmento Creado	0	0	0	0	…
Segmento Asignado	1	0	0	0	…
	-1	0	0	0	Ya fue actualizado

***Ilustración 9**: Cálculo de Impacto en las restricciones por segmento*

Los valores obtenidos se suman a la función objetivo global, actualizando las violaciones individuales y de segmento.

- **Restricciones a nivel de la solución:** Se reemplaza el segmento asignado por el creado y se calculan las restricciones Dco, Dsc, Dsa, Sco, y dci. A estos valores se restan las violaciones registradas en la función objetivo global, teniendo en cuenta que las restricciones individuales y por segmento han sido actualizadas previamente.

	Dco	Dsc	Dsa	Sco	dci	Restricciones segmento	Restricciones individuales
F.O. de la nueva solución	0	0	1	0	5	…	…
F.O. de la solución anterior	2	0	3	0	3	…	…
	-2	0	-2	0	2	Ya fue actualizado	Ya fue actualizado

Ilustración 10: *Cálculo del impacto en las restricciones globales*

La Ilustración 11 presenta el diagrama de clases de los objetos que intervienen en la medición del impacto. Cada clase posee un atributo función objetivo donde se almacenan las diferentes violaciones. El cálculo de las restricciones a nivel individual se guarda en la clase Movimiento con la que se actualizará la función objetivo del nuevo segmento. En la clase Segmento se guardan las violaciones por segmento y se actualiza la función global de Candidato, para finalmente evaluar las restricciones globales y tener la totalidad de las violaciones en esta clase. Con este proceso se evita el tener que revisar todas las asignaciones de la solución y se obtienen los valores respectivos de F.O.

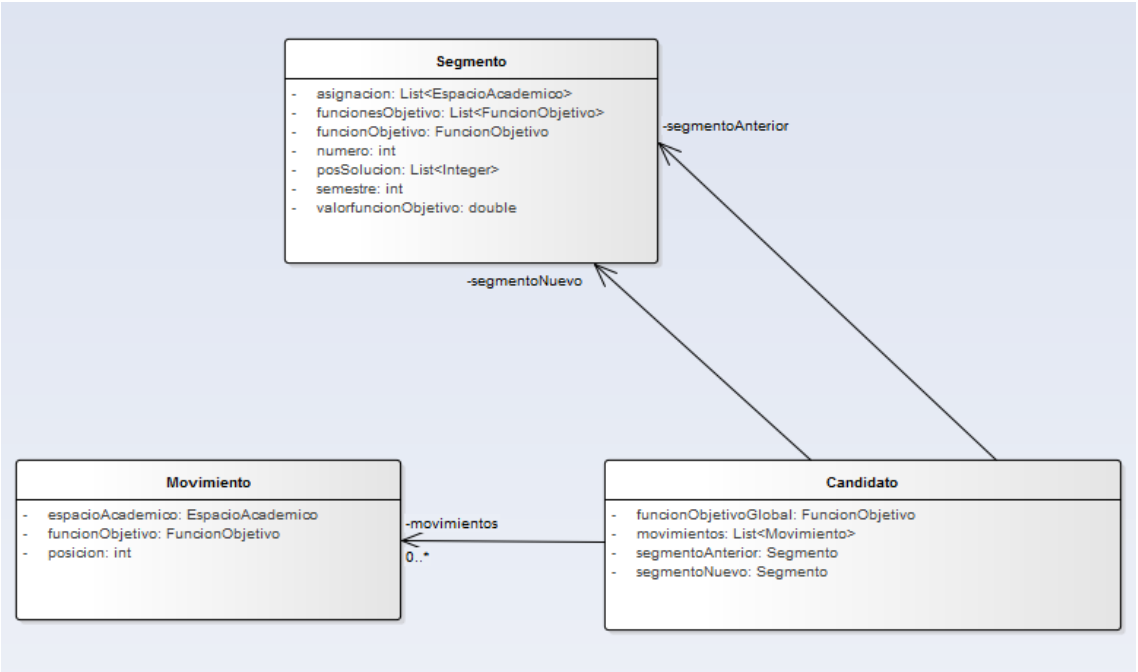


Ilustración 11: *Objetos que intervienen en la medición del impacto a la función objetivo*

4.2.2.3 *Metodología de Asignación*

Basado en un número máximo de iteraciones, búsqueda tabú intenta mejorar la asignación entregada por el algoritmo voraz. De acuerdo con la fuerza de perturbación se seleccionan aleatoriamente β grupos, y para cada uno ellos se buscan α candidatos, teniendo en cuenta los elementos que pertenecen a la lista tabú. Cada candidato está conformado por uno o varios movimientos, dependiendo de la cantidad de espacios que se deben asignar por grupo. Una vez se tienen dichos candidatos, se busca cuál de ellos arroja el menor valor de función objetivo y se incluye dentro de la solución.

Se define una numeración por grupo para realizar las validaciones en el algoritmo. Al primer grupo de cada asignatura se le asigna el número 0, al segundo el 1, al tercero el 2 y así sucesivamente. De esta forma los primeros grupos de todas las asignaturas del semestre conformarán el segmento de horario 0, los segundos el segmento 1 etc.

Dicha numeración denominada "segmentos de horario" solo se aplica en el cálculo de la función objetivo y, permite agrupar espacios académicos a fin de garantizar que el estudiante tenga al menos una opción para tomar las asignaturas de su respectivo semestre. De esta forma se permite la creación de un horario compacto y que evite los cruces entre los grupos de un mismo segmento.

Se mantienen 2 soluciones activas mientras el algoritmo está en ejecución: la primera es la solución sobre la que se realizan las modificaciones (solución actual) y la segunda corresponde a aquella con la mejor calidad o el menor valor de función objetivo (mejor solución).

Después de construir la lista de candidatos, se selecciona aquel que posee la menor función objetivo o que genera el menor impacto en la solución. Sin importar el segmento al que pertenece el grupo, o si mejora la solución actual, los movimientos son realizados y el grupo es agregado a la lista tabú. Cuando el candidato seleccionado reduce el valor de la función objetivo, la solución actual se convierte en la mejor solución y los valores de α y β regresan al valor inicial. Si no se mejora la función objetivo, se incrementa el valor de β para diversificar la búsqueda, pero sino es posible obtener algún candidato se incrementa el valor de α. Cuando el candidato no se consigue una mejora, se incrementa el contador de iteraciones sin mejora ξ. Una vez que ξ alcanza el número máximo, la mejor solución se convierte en la solución actual, empezado de nuevo la búsqueda a partir del horario con la mayor calidad encontrada hasta el momento.

Si el mejor candidato se encuentra marcado como tabú deberá superar el criterio de aspiración. Este determina que para seleccionar dicho elemento, debe minimizar la función objetivo de la mejor solución encontrada hasta ese momento. Si el mejor candidato es tabú y no supera el criterio de aspiración se seleccionará el siguiente mejor candidato, pero en caso contrario, el grupo para el que se obtuvo el candidato entrará nuevamente a lista tabú.

En la lista tabú pueden existir grupos repetidos, teniendo en cuenta que el número de apariciones no se toma en cuenta bajo ninguna circunstancia; simplemente se realiza la validación de la existencia de un grupo en la lista tabú para verificar si supera el criterio de aspiración. Los elementos repetidos irán saliendo de la lista a medida que avanza el algoritmo, tal y como ocurre con el resto de grupos.

Teniendo en cuenta las variables mencionadas al inicio de la sección 4 y la metodología presentada anteriormente, la Ilustración 10 presenta el pseudocódigo del algoritmo ATS implementado.

Variable	Descripción
X_{mejor}	Mejor solución
X_{actual}	Solución actual
α	Profundidad
α_{inc}	Incremento de la profundidad
α_{min}	Profundidad mínima
α_{max}	Profundidad máxima
β	Fuerza de perturbación
β_{inc}	Incremento de la fuerza de perturbación
β_{max}	Máximo valor de la fuerza de perturbación
β_{min}	Valor Mínimo de la fuerza de perturbación
ξ	Iteraciones sin mejora
ξ_{max}	Máximo número de iteraciones sin mejora
L_G	Lista de Grupos para los que se deben generar candidatos. Su tamaño está determinado por β
G	Grupo
L_C	Lista con todos los candidatos obtenidos a partir de L_G. Su tamaño está determinado por $\alpha * \beta$
$L_{E(G)}$	Lista de espacios académicos que pertenecen al grupo G
$\lvert L_{E(G)} \rvert$	Número de espacios que pertenecen al grupo G
M_n	Movimiento número n
$E_n(G)$	Espacio número n que pertenece al grupo G
Ci_G	Candidato número i para el grupo G
Fi_M	Función objetivo con las restricciones individuales del movimiento M
$f_i(M, E)$	Cálculo de las restricciones individuales violadas por el movimiento M al reemplazar el espacio E. (Ilustración 8)
Fz_C	Función objetivo con las restricciones de segmento e individuales si se selecciona el Candidato C
Z_G	Segmento al que pertenece el grupo G
$f_z(C, Z)$	Cálculo de las restricciones por segmento violadas al reemplazar el segmento Z por el creado con los movimientos del candidato C. (Ilustración 9)
Fo_C	Función objetivo (con todas las restricciones violadas) si se selecciona el candidato C
$F_{Xactual}$	Función objetivo de la solución actual (todas las restricciones)
F_{Xmejor}	Función objetivo de la mejor solución (todas las restricciones)
$f_o(C, F)$	Cálculo de todas las restricciones violadas al realizar los movimientos del candidato c. (Ilustración 10)

G_C	Grupo para el que se creó el candidato C
ℸ	Lista Tabú
A_C	Mejor candidato en L_C
Fo_{Ac}	Función objetivo del mejor candidato (con todas las restricciones)
G_{Ac}	Grupo para el que se generó el mejor candidato
σ	Número máximo de iteraciones

***Tabla 2**: Descripción de variables usadas en el algoritmo F-ATS*

Búsqueda Tabú

X_{mejor} ← solución Inicial
X_{actual} ← X_{mejor}
α ← α_{min}, β ← β_{min}, ξ ← 0
Repetir
L_G ← obtener aleatoriamente β grupos
Para cada G en L_G *// obtener* L_C
Desde i = 0 hasta α
Desde n = 0 hasta n < $|L_{E(G)}|$
Crear M_n para reemplazar a $E_n(G)$
Agregar M_n a Ci_G
Fi_{Mn} ← $f_i(M_n, E_n(G))$
Actualizar Fz_{CiG} con los valores Fi_{Mn}
Agregar Ci_G a L_C
Fz_{CiG} ← $f_z(Ci_G, Z_G)$
Actualizar Fo_{CiG} con los valores Fz_{CiG}
Fo_{CiG} ← $f_o(Ci_G, F_{Xactual})$
//fin Desde
//fin Para
A_C← Mejor candidato en L_C *// si* G_{Ac} ϵ ℸ *debe superar el* ***criterio de aspiración***
Si A_C <> null
Ejecutar A_C en X_{actual} *// se reemplazan los movimientos de* A_C
$F_{Xactual}$ ← Fo_{Ac}
Agregar $G(A_C)$ a ℸ
Si $F_{Xactual} < F_{Xmejor}$ *// también es usado como* ***criterio de aspiración***
X_{mejor} ← X_{actual}
F_{Xmejor} ← $F_{Xactual}$
α ← α_{min}, β ← β_{min}, ξ ← 0
Sino
ξ ← ξ+1
β ← Menor(β + β_{inc} , β_{max}) *// **diversificación***
Sino
ξ ← ξ+1
α ← Menor(α + α_{inc} , α_{max}) *// **intensificación***

Si ξ = ξ_{max}
X_{actual} ← X_{mejor} *// reinicio de la búsqueda*
$F_{Xactual}$ ← F_{Xmejor}

hasta alcanzar σ ó $F(X_{actual}) = 0$

Fin Búsqueda Tabú

***Ilustración 12:** Estructura del algoritmo de búsqueda tabú adaptativa implementado, fuente propia.*

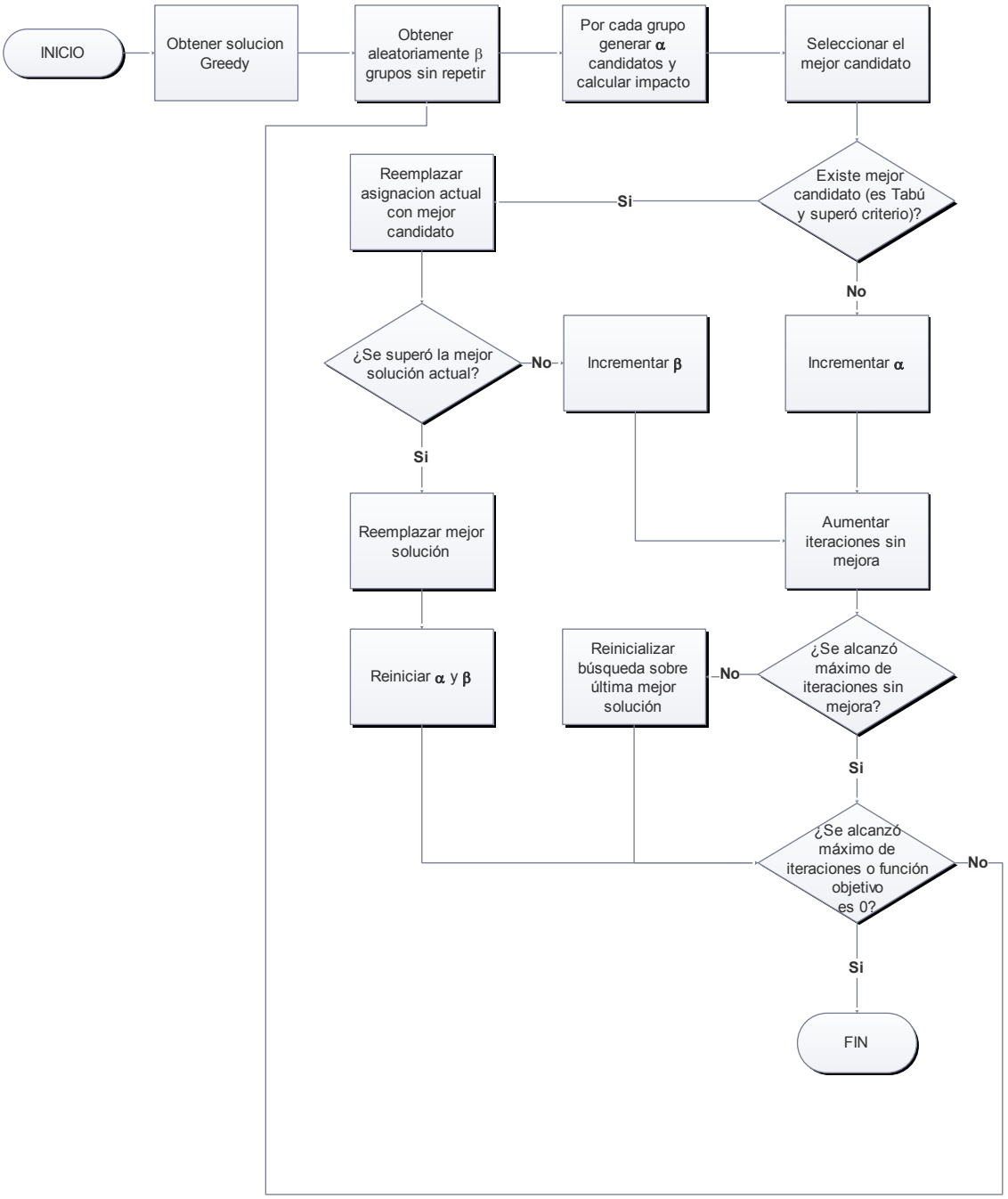


***Ilustración 13**: Diagrama de flujo algoritmo ATS implementado*

4.2.2.4 *Estructuras Utilizadas*

- Lista tabú: Cola en la que se irán almacenando los grupos para los que se han realizado movimientos. Su tamaño está determinado por el valor del parámetro "tamaño de lista tabú".
- Carga de docentes y carga de salones: Lista con la carga de cada uno de los docentes/salones en el proceso de asignación. Permite determinar de forma rápida aquellos docentes/salones candidatos a incluir en un movimiento.

- Listado de funciones objetivo: Lista con la función objetivo asociada a cada uno de los espacios existentes en la solución. Es del mismo tamaño que la lista total de espacios académicos creados.
- Listado de segmentos: lista que contendrá los segmentos.
- Lista de candidatos: Lista que contiene los candidatos de todos los grupos escogidos en una iteración. Esta lista se crea nuevamente con cada iteración.

4.2.2.5 *Configuración de Parámetros*

Debido a que el número de parámetros utilizados en el algoritmo implementado es considerable, no es posible recurrir al método de "grid search" (Kim, 1997) para determinar la configuración con la que ATS obtendría mejores resultados ante el problema propuesto. Por tal razón, se recurrió al análisis de la varianza o ANOVA, que permite determinar si una o más medidas presentan diferencias significativas respecto a un factor (M. Lane, 2011).

PARÁMETRO	RESTRICCIÓN
β_{min}	Valor mínimo de fuerza de perturbación
β_{max}	Valor máximo de fuerza de perturbación
β_{nc}	Incremento de fuerza de perturbación
α_{min}	Valor mínimo de profundidad
α_{max}	Valor máximo de profundidad
α_{inc}	Incremento de profundidad
ξ_{max}	Máximo de Iteraciones sin mejora
$\lvert T \rvert$	Tamaño de la lista tabú

***Tabla 3**: Parámetros del algoritmo ATS implementado*

Existen diferentes variantes de ANOVA, la que se ha utilizado en este desarrollo se conoce "ANOVA unidireccional", que básicamente "implica el análisis de datos muestreados de más de dos poblaciones (distribuciones) numéricas o de datos de experimentos en los cuales se utilizaron más de dos tratamientos. La característica que diferencia los tratamientos o poblaciones una de otra se llama factor en estudio y los distintos tratamientos o poblaciones se conocen como niveles del factor" (Devore, 2008). Siguiendo el planteamiento realizado en otros trabajos ((Bell & McMullen, 2004), (Dengiz & Alaba, 2000), (Kolahan & Doughabadi, 2012), (Pertuz Montenegro & Rojas Silva, 2007), (Saremi, ElMekkawy, & Wang, 2007)) el factor de estudio corresponde a la función objetivo mientras que los niveles son los diferentes parámetros del algoritmo.

ANOVA determina si existen parámetros significativos respecto a la función objetivo, pero para determinar cuáles son, es necesario el uso de algún test *Pos Hoc* (a posteriori). En este desarrollo se implementó el test de Tukey (Devore, 2008), que determina

cuales de los niveles del factor son los más relevantes, con base en los resultados de ANOVA.

4.2.2.5.1 Primera Etapa

Para determinar la configuración de ATS a través de ANOVA, se definieron los siguientes "Set's de Prueba" (Conjunto de valores que tomará cada parámetro). Cada Set varia un parámetro con diferentes valores establecidos de forma arbitraria, mientras se mantiene el resto en un valor base. De esta manera se podrá medir la varianza de cada parámetro en la Tabla 3 con respecto a la función objetivo:

	Base	Min Perturbación	Max Perturbación	Inc Perturbación	Min Profundidad	Max Profundidad	Inc Profundidad	Ite Mejora	tam tabú
βmin	5	**2, 3, 5, 7, 10**	5	5	5	5	5	5	5
βmax	10	10	**12, 13, 15, 17, 20**	10	10	10	10	10	10
βinc	1	1	1	**2, 3, 5, 7, 10**	1	1	1	1	1
αmin	5	5	5	5	**10, 13, 15, 17, 23**	5	5	5	5
αmax	10	10	10	10	10	**13, 15, 17, 23, 27**	10	10	10
αinc	2	2	2	2	2	2	**1, 3, 5, 7, 10**	2	2
ξmax	10	10	10	10	10	10	10	**13, 15, 17, 20, 23**	10
| τ |	10	10	10	10	10	10	10	10	**13, 15, 17, 20, 23**

***Tabla 4**: Set's de Prueba para determinar los parámetros significativos de ATS*

Para el cálculo de ANOVA y del Test de Tukey se hizo uso del software MINITAB[5] y cada uno de los Set's de prueba se ejecutó 50 veces para cada valor establecido (250 muestras por parámetro).

[5] Software estadístico comercial para ejecutar funciones avanzadas. Se hizo uso de la versión de prueba para realizar los cálculos.
Disponible en http://it.minitab.com/es-mx/products/minitab/free-trial.aspx/default.aspx

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Factor	7	860042	122863	93,98	0,000
Error	792	1035360	1307		
Total	799	1895403			

Ilustración 14: *Tabla ANOVA para ATS. Obtenida con el software MINITAB*

Para que existan variables relevantes el valor P-Value debe ser menor a 0.05, que corresponde al error estándar permitido. La Ilustración 14 presenta los resultados del análisis de la varianza para ATS y permite afirmar que existen parámetros significativos.

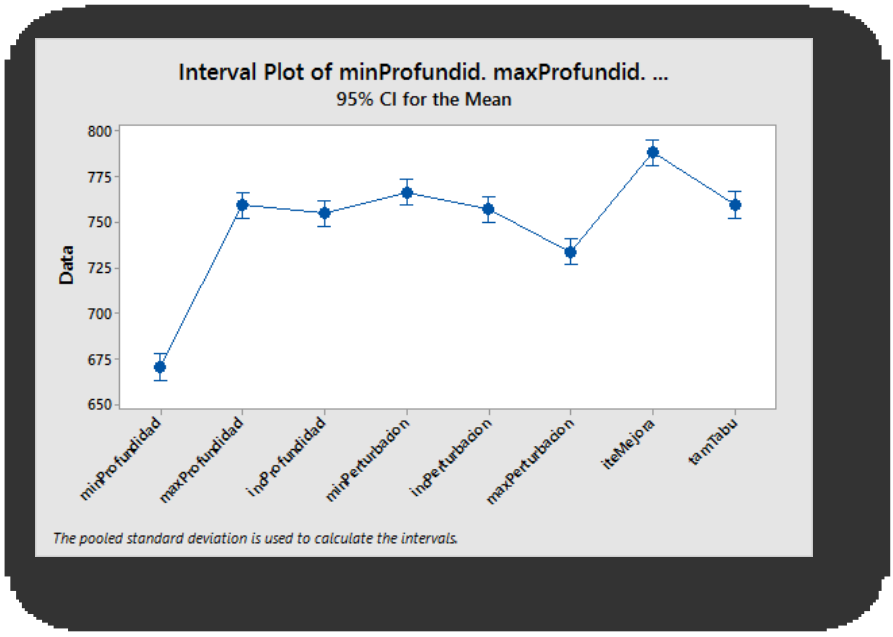


Ilustración 15: *Varianza de cada parámetro de ATS. Obtenida con el software MINITAB*

La Ilustración 15 presenta la varianza de cada parámetro con respecto a la función objetivo. Aunque se puede observar que el valor α_{min} tiene gran relevancia, el resto de parámetros presentan similitudes. El test de Tukey permite identificar con la misma certeza utilizada en ANOVA (95%) cuales de los parámetros son significativos:

Tukey Pairwise Comparisons

Grouping Information Using the Tukey Method and 95% Confidence

Factor	N	Mean	Grouping
iteMejora	250	788,48	A
minPerturbacion	250	766,44	B
tamTabu	250	759,63	B
maxProfundidad	250	759,48	B
incPerturbacion	250	757,19	B
incProfundidad	250	755,00	B
maxPerturbacion	250	733,70	C
minProfundidad	250	670,40	D

Means that do not share a letter are significantly different.

***Ilustración 16**: Test de Tukey para ATS. Obtenida con el software MINITAB*

El *test de Tukey* que arroja MINITAB clasifica con una variable cada uno de los parámetros utilizados, aquellos que no comparten una letra son significativamente diferentes. Con base en los resultados de la Ilustración 16 los parámetros más relevantes en el algoritmo ATS son: ξ_{max}, α_{min} (tal y como se observa en la Ilustración 15) y β_{max}.

Estos resultados son congruentes con el funcionamiento de ATS, valores altos de α y β generan una mayor exploración por parte del algoritmo. Con cada incremento de β, ATS se acerca a explorar la totalidad de grupos registrados en la solución, y si α crece, se buscará un mayor número de candidatos que mejoren las asignaciones de un grupo. Dado que β_{max} resultó ser el parámetro significativo relacionado con la fuerza de perturbación, se puede afirmar que el mecanismo de diversificación se ejecuta con gran frecuencia; el valor de β_{max} es alcanzado un gran número de veces por lo que tiene mayor relevancia que los parámetros β_{min} y β_{inc}. Esto no ocurre con la profundidad, dado que el mecanismo de intensificación depende de la incapacidad para hallar un candidato, y se tiene una mayor probabilidad de encontrarlo (aunque este empeore la solución), α no varía con tanta frecuencia como β. Por tal razón, el valor de la profundidad está determinado básicamente por α_{min}, superando en relevancia a α_{inc} y α_{max}.

ATS selecciona el candidato que posee la menor función objetivo dentro del listado de candidatos, por lo que no se tiene garantía de que el elemento seleccionado produzca una mejor solución; de esta manera, la solución actual puede ir empeorando y llevando al algoritmo a estancarse en un óptimo local. El parámetro de iteraciones sin mejora permite que ATS escape de estos estancamientos y continúe con el proceso de búsqueda. La información obtenida en el *test de Tukey* permite corroborar la gran importancia de ξ_{max} para permitir que la búsqueda no quede atrapada en un óptimo local.

4.2.2.5.2 Segunda Etapa

Una vez obtenidos los parámetros relevantes para el algoritmo ATS, se realiza un pequeño *grid search* con el que se determinan aquellos valores de ξ_{max}, α_{min} y β_{max} que en conjunto generan mejores soluciones. Para la realización de estas pruebas se hizo uso de una herramienta llamada "Perfidix"[6], y al igual que con los datos utilizados para el cálculo de ANOVA, se definieron Set's de prueba ejecutados 50 veces.

	Valor 1	Valor 2	Valor 3	Valor 4	Valor 5
ξ_{max}	5	7	10	15	21
α_{min}	3	5	9	15	21
β_{max}	5	7	10	15	21

***Tabla 5**: Valores de parámetros significativos de ATS para la ejecución de Grid Search*

Los valores en la Tabla 5 fueron establecidos de forma arbitraria, a excepción de α_{min} para el que se tomaron 3, 5 y 9 con el fin de verificar si con una profundidad pequeña se pueden obtener buenas soluciones. El resto de parámetros fue configurado con los valores base de la Tabla 3.

Con base en la solución entregada por la universidad (sección 4.2.5.3) se normaliza el cálculo de la función objetivo tal como se muestra en la Ecuación 45, de tal manera que los valores que podrá tomar la función se encontraran en el rango [0,1].

$$FN_S = \frac{FO_S}{FO_U}$$

***Ecuación 45**: Normalización de la función objetivo. Fuente propia.*

Donde FO_S es el valor obtenido por una solución S y FO_U es la función objetivo de la solución entregada por la universidad, teniendo en cuenta que ambos cálculos se realizan con base en la *Ecuación 5*.

A partir de la combinatoria de los valores en la Tabla 5, se generaron un total de 125 Set's de prueba. A continuación se presentan los 5 mejores resultados:

[6] Es una herramienta que permite hacer BenchMarks sobre código JAVA. Se optó por usar esta herramienta ya que a diferencia de JUnit (Framework para pruebas unitarias en JAVA) se pueden definir diferentes métricas (tiempo, consumo de memoria y métricas personalizadas) y obtener un gran número de datos (promedio, desviación estándar e intervalo de confianza entre otros.)

Parámetro			FO_S		FN_S	
ξ_{max}	α_{min}	β_{max}	Promedio	Desviación	Promedio	Desviación
5	21	21	644.22	32.38	0.3650	0.0183
5	21	15	647.22	36.64	0.3667	0.0208
7	21	21	647.80	32.54	0.3670	0.0184
7	21	15	650.74	30.21	0.3687	0.0171
10	21	21	655.12	33.31	0.3712	0.0189

***Tabla 6**: Mejores resultados del Grid Search para el algoritmo ATS*

Como se puede observar en los resultados obtenidos, con valores altos de α y β se obtienen menores valores en la función objetivo. Tal y como se explicaba anteriormente, al incrementar la profundidad y la fuerza de perturbación se crea un mayor número de candidatos, pero se genera un incremento en el tiempo de respuesta y ATS empieza a comportarse como un algoritmo voraz (intentando explorar todas las soluciones posibles, razón por la cual no se incrementaron más los valores de α_{min} y β_{max}). También se hace notorio que ξ_{max} favorece al algoritmo cuando tiene un valor más pequeño ya que con un número de menor de iteraciones sin mejora, ATS escapará más rápido de óptimos locales.

El primer Set en la Tabla 6 presenta el mejor promedio y la segunda desviación estándar más pequeña, por tal razón la configuración del algoritmo ATS que será usada en las comparaciones será la siguiente:

Parámetro	Valor
β_{min}	5
β_{max}	**21**
β_{nc}	1
α_{min}	**21**
α_{max}	30
α_{inc}	2
ξ_{max}	**5**
$\lvert T \rvert$	10

***Tabla 7:** Configuración final del algoritmo ATS*

4.2.2.5.3 Tendencia del algoritmo

Antes de realizar la comparativa de los algoritmos, se realiza un análisis del comportamiento de ATS respecto al número de iteraciones. Para tal fin se ejecutaron pruebas cada 100 iteraciones, hasta llegar a un máximo de 3000. Los resultados se presentan a continuación:

ITERACIONES	Promedio de FN	Desviación de FN	Diferencia con el resultado anterior
0	0.6023	0.0366	------
100	0.4668	0.0273	0.1355
200	0.4165	0.0182	0.0502
300	0.4007	0.0158	0.0159
400	0.3887	0.0170	0.0120
500	0.3775	0.0188	0.0112
600	0.3640	0.0189	0.0135
700	0.3639	0.0189	0.0001
800	0.3568	0.0190	0.0071
900	0.3526	0.0170	0.0042
1000	0.3443	0.0202	0.0083
1100	0.3424	0.0175	0.0019
1200	0.3421	0.0168	0.0003
1300	0.3380	0.0202	0.0041
1400	0.3372	0.0183	0.0008
1500	0.3342	0.0182	0.0030
1600	0.3332	0.0183	0.0010
1700	0.3340	0.0152	-0.0007
1800	0.3337	0.0147	0.0002
1900	0.3247	0.0178	0.0091
2000	0.3292	0.0185	-0.0045
2100	0.3291	0.0189	0.0001
2200	0.3248	0.0158	0.0043
2300	0.3281	0.0171	-0.0033
2400	0.3206	0.0178	0.0076
2500	0.3269	0.0185	-0.0063
2600	0.3206	0.0182	0.0063
2700	0.3233	0.0168	-0.0027
2800	0.3186	0.0178	0.0047
2900	0.3167	0.0165	0.0019
3000	0.3225	0.0159	-0.0058

***Tabla 8**: Resultados de las pruebas de tendencia para ATS*

La Tabla 8 presenta los promedios y las desviaciones estándar obtenidos con cada número de iteraciones, en la última columna se presenta la diferencia del promedio respecto al resultado de la prueba anterior. Al analizar esta información se puede ver como el valor de la función objetivo se va reduciendo a medida que el número de

iteraciones crece, y que conforme aumenta el número de iteraciones, la mejora en la función objetivo se reduce (si se observan las diferencias de los resultados en la tercera columna, es notorio que dicha diferencia se va haciendo cada vez más pequeña). Para facilitar el análisis de la tendencia de ATS se presenta la siguiente gráfica:

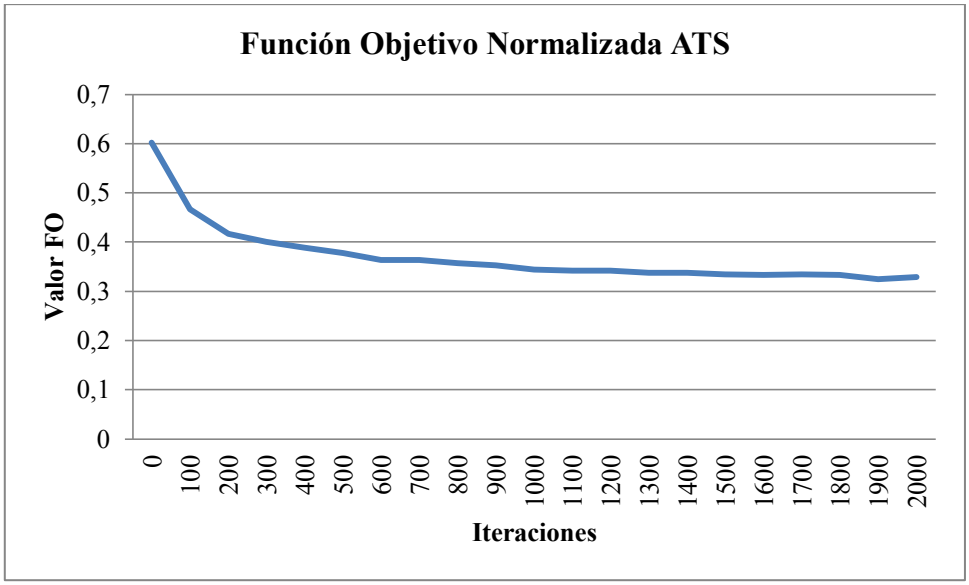


***Ilustración 17**: Tendencia del algoritmo ATS*

Entre las iteraciones 0 y 300 se puede ver una pendiente bastante marcada, indicando que ATS reduce rápidamente la función objetivo de la solución entregada por el algoritmo greedy. Con un mayor número de iteraciones se producen mejores soluciones, pero tal y como se observó en la tabla de resultados, esta mejora resulta ser cada vez menor. Por esta razón la gráfica parece estabilizarse después de las 2000 iteraciones.

El algoritmo presenta una tendencia notoria a mejorar la función objetivo, pero se observan algunos picos en los resultados obtenidos. Pese a que cada prueba se ejecutó 50 veces no siempre se mejoran los resultados de la prueba anterior, razón por la que se observan valores negativos en la Tabla 8.

4.2.3 ALGORITMOS GENÉTICOS

La estructura para la implementación de la técnica, se basó en el planteamiento del artículo de (Phuc, Khang, & Nuong, 2011) haciendo ajustes en la representación, para tomar como población inicial la solución entregada por el algoritmo voraz descrito en este documento en la sección de "Implementación de las Técnicas".

4.2.3.1 *Evaluación de la Función Objetivo*

La función objetivo en algoritmos genéticos se evalúa por cada individuo, en su creación o en el uso de algún operador genético (cruce o mutación), así, mediante la selección compararlos y elegir la siguiente generación.

4.2.3.2 *Metodología de Asignación*

Para el algoritmo genético, una posible solución se obtiene por medio de una población inicial y los operadores (Selección, cruce, mutación, etc.), a continuación se describen sus implementaciones:

Población Inicial, al inicio se genera con ayuda de un Algoritmo Voraz en esta caso Greedy un número de individuos para obtener diversificación.

Operador de Selección, mantendrán un número de individuos por cada iteración, haciendo uso de la selección por torneo.

Operador de cruce, hace el intercambio de material genético de forma uniforme, validando la factibilidad y la mejora en la función objetivo respecto al padre, si no hace una mejora y no es factible no hace el intercambio.

Operador de Mutación, el gen a mutar se elige aleatoriamente de la lista de posibles opciones, de modo que el cromosoma (posible solución) sea factible, cambiando la franja horaria y el salón del objeto de espacio académico.

Lo mencionado se observa en el pseudocódigo de la Ilustración 18 con su descripción de las variables (Ver Tabla 9).

Variable	Descripción
X	Individuo o cromosoma
X_{mejor}	Mejor individuo
X_{padre}	Individuo padre
X_{hijo}	Individuo hijo
ρ	Probabilidad de selección
ρ_{cruce}	Probabilidad para realizar el operador de cruce
$\rho_{mutación}$	Probabilidad para realizar el operador de mutación
P	Población
σ	Máximo número de iteraciones del algoritmo
Gen	Gen de un cromosoma x
f(X)	Evaluación de la función objetivo de un individuo x
mutar(X)	Función que muta un individuo x
intercambio(Gen_1, Gen_2)	Función que hace el intercambiando de material genético entre dos genes *Gen,*.
selTorneo(P)	Función que hace la selección de un individuo x por Torneo de una población P
selMejorX(P)	Función que busca el mejor individuo X_{mejor} de una población P
genProb()	Función que genera una probabilidad (número aleatorio generado de 0 a 100).
genPobInicial()	Función que genera la población inicial a partir del algoritmo Greedy
selSiguienteGen(P)	Función que construye la generación siguiente.

***Tabla 9**: Variables usadas en el pseudocódigo.*

```
Algoritmo genético
        P ← genPobInicial ()
        X_mejor ← selMejorX (P)
        Mientras contador < σ y f(X_mejor) > 0 hacer
                X_padre1 ← selTorneo (P)
                X_padre2 ← selTorneo (P)

                ρ ← genProb()
                Si ρ < ρ_cruce entonces
                        //Inicio Cruce
                        X_hijo1 ← X_padre1
                        Desde i=0 hasta número máximo de genes de X_padre1
                                Gen_1 ← Obtener Gen_i de X_padre1
                                Gen_2 ← Obtener Gen_i de X_padre2
                                Gen_hijo ← intercambio (Gen_1 ,Gen_2)
                                Si f (Gen_hijo U X_hijo1) < f (X_hijo1) entonces
                                        X_hijo1. Gen_i ← Gen_hijo
                                Fin si
                        Fin desde
                        X_hijo2 ← X_padre2
                        Desde i=0 hasta número máximo de genes de X_padre2
                                Gen_1 ← Obtener Gen_i de X_padre1
                                Gen_2 ← Obtener Gen_i de X_padre2
                                Gen_hijo ← intercambio (Gen_2 ,Gen_1)
                                Si f (Gen_hijo U X_hijo2) < f (X_hijo2) entonces
                                        X_hijo2. Gen_i ← Gen_hijo
                                Fin si
                        Fin desde
                        //Fin Cruce
                Si no
                        X_hijo1 ← X_padre1
                        X_hijo2 ← X_padre2
                Fin si
                ρ ← genProb()
                Si ρ < ρ_mutación entonces

                        Para cada X_hijo i
                                X_hijo i ← mutar (X_hijo i)
                                f(X_hijo i)
                        Fin para
                Fin si
                P ← P + X_hijo 1 + X_hijo 2
                P ← selSiguienteGen( P)
                X_mejor ← selMejorX (P)
                contador ← contador +1
        Fin Mientras
        Retornar X_mejor
Fin
```

***Ilustración 18**: Estructura en seudocódigo del algoritmo genético Implementado, fuente propia.*

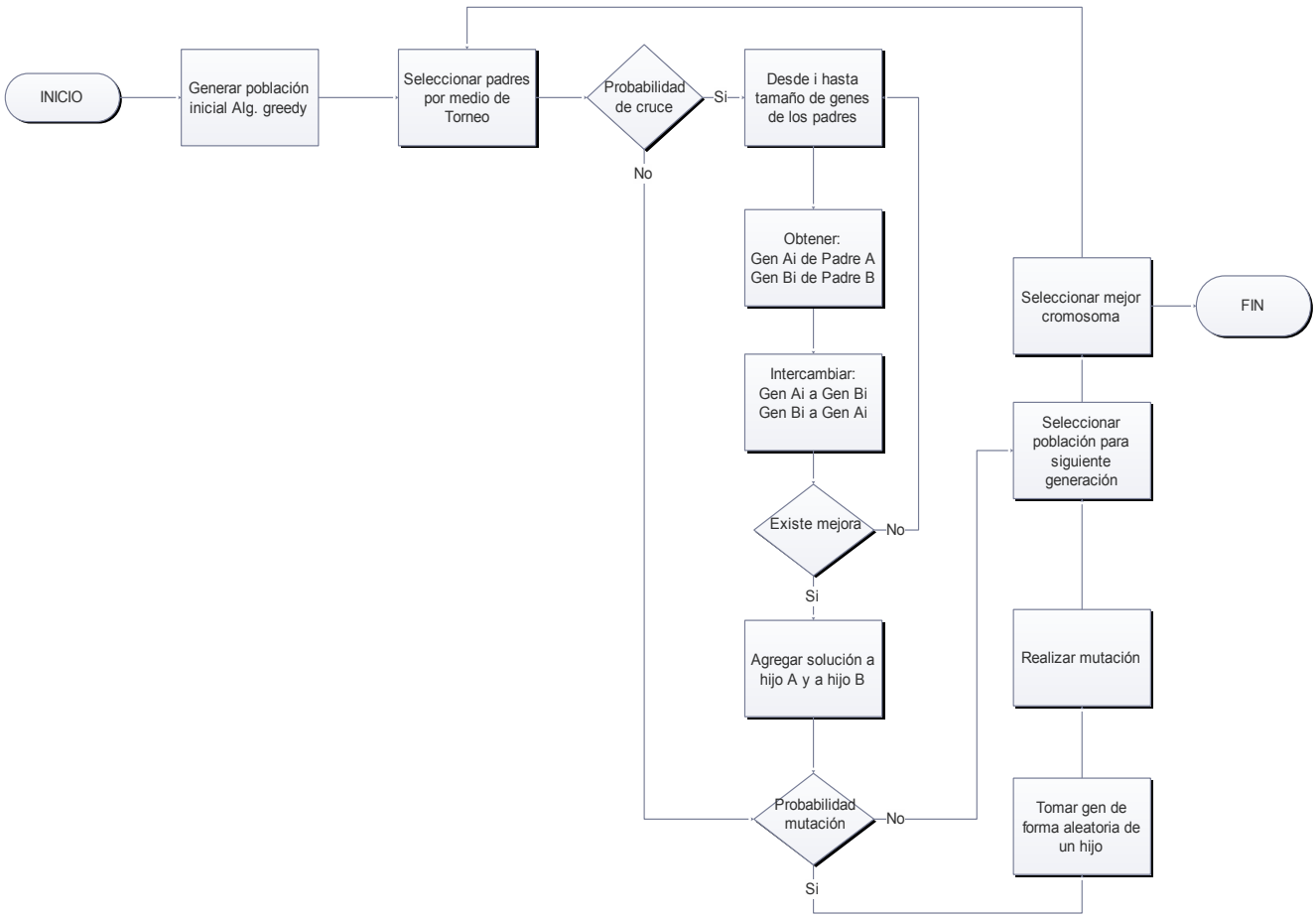


Ilustración 19: Diagrama de flujo de Algoritmo Genético implementado

4.2.3.3 ***Estructuras Utilizadas***

En la literatura consultada ((Melanie, 1998), (Chacón Montes, 1995), (Mendez Giraldo, Caballero Villalobos, & Álvarez Pomar, 2007) entre otros) es frecuente el uso del planteamiento binario, inicialmente se abordó esta estructura, pero debido a las variables, restricciones (enunciadas en la sección "Modelamiento de la solución") y al uso del algoritmo Greedy como suministro de la población inicial, los tiempos de ejecución se hacían extensos a medida que se agregaban más individuos para obtener alguna mejora, esto debido a la creación de varias matrices, que representan las relaciones de cada variable con su respectiva restricción y hacer la conversión de objeto (que es lo que entrega el algoritmo Greedy) a matrices binarias. De modo que se optó por la representación siguiente:

Un cromosoma o posible solución, se representa como una lista de objetos de espacio académico.

***Ilustración 20**: Ejemplo de un gen para la implementación del algoritmo genético*

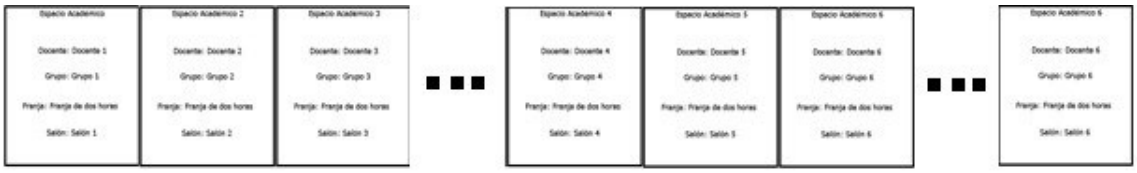


***Ilustración 21**: Ejemplo de un cromosoma o posible solución para la implementación del algoritmo genético*

4.2.3.4 *Configuración de Parámetros*

Como fue mencionado en Algoritmo Tabú, para determinar la configuración del Algoritmo Genético, primero se determinará si existen parámetros significativos usando ANOVA y luego con *grid seach* obtener los valores que deben tomar para generar el mejor resultado en la función objetivo.

4.2.3.4.1 Primera Etapa

Se definieron un conjunto de valores que tomará cada parámetro ("*Set's* de prueba"), variándolos de forma arbitraria por cada conjunto *("Set")*. El *Set Base* es la prueba de control, permite ver el impacto en el resultado cuando se modifica cada parámetro de esta manera poder medir la varianza de cada uno.

VARIABLES	*Set* Base	*Set* 1	*Set* 2	*Set* 3	*Set* 4
Número de genes a cruzar	100	150, 200, 300, 400	100	100	100
Población Inicial	100	100	10, 50, 150, 200	100	100
Probabilidad de Mutación	10	10	10	10, 30, 80, 100	40
Probabilidad de Cruce	60	60	60	30	1, 20, 40 100

***Tabla 10**: Set's de pruebas usados para determinar los parámetros significativos en el Algoritmo Genético.*

Cada uno de los Set´s de pruebas se ejecutó 50 veces, haciendo uso de la aplicación MINITAB para el cálculo de ANOVA y Test de Turkey.

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Factor	3	923036	307679	113,03	0,000
Error	796	2166708	2722		
Total	799	3089744			

***Ilustración 22**: Tabla ANOVA para Algoritmo Genético. Obtenida con el software MINITAB*

En la ilustración anterior muestra que el error estándar permitido (P-Value en MINITAB) está debajo de 0.05, con esto se puede afirmar que existen parámetros significativos.

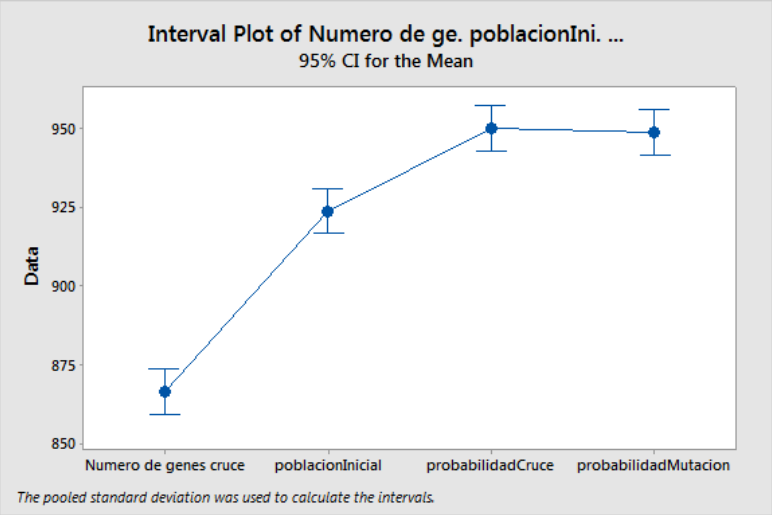


***Ilustración 23**: Varianza de cada parámetro de Algoritmo Genético. Obtenida con el software MINITAB*

La anterior ilustración presenta la varianza de cada parámetro de Algoritmo Genético con respecto a la función objetivo, identificando que el número de genes de cruce tiene gran relevancia con respecto a los demás.

Tukey Pairwise Comparisons

Grouping Information Using the Tukey Method and 95% Confidence

Factor	N	Mean	Grouping
probabilidadCruce	200	950,29	A
probabilidadMutacion	200	949,01	A
poblacionInicial	200	923,88	B
Numero de genes cruce	200	866,48	C

Means that do not share a letter are significantly different.

***Ilustración 24**: Test de Turkey para Algoritmo Genético. Obtenida con el software MINITAB*

El Test de Turkey de MINITAB muestra las variables significativas usando letras, aquellas que comparten letra no lo son.

Mediante la anterior ilustración se puede determinar, que esta implementación del Algoritmo Genético las variables relevantes son: Población inicial y el Número de genes de cruce. Esto es coherente ya que a mayor número de individuos se puede obtener mayor diversidad y así al momento del uso de los operadores genéticos no tener una convergencia prematura a un óptimo local, el Número de genes de cruce es una oportunidad de mejora de la función objetivo, debido a que el cruce entre genes, solo se hace si el resultado que se genera tiene mejor función objetivo que el padre, entonces a mayor oportunidad (valor) de cruce, mayor será la mejora, en cuanto a los otros parámetros. La probabilidad de mutación no fue relevante, ya que a medida que se le da mayor valor, se incrementa la función objetivo y el resultado va ser un individuo generado al inicio dentro de la población inicial. La probabilidad de cruce debe de ir de la mano del número de genes para el cruce, por si sola no hace una operación para mejorar la función objetivo.

4.2.3.4.2 Segunda Etapa

Ya con las variables relevantes se procede a hacer *grid search* para obtener los valores que en conjunto generen un mejor valor de función objetivo. Al igual que los datos utilizados para el cálculo del ANOVA, se definieron Set´s de pruebas ejecutados 50 veces. Para la prueba se hizo uso de la herramienta "*Perfidix*" y con un valor de iteración de 100.

VARIABLES	*Valor* 1	*Valor* 2	*Valor* 3	*Valor* 4
Número de genes a cruzar	100	200	300	400
Población Inicial	100	200	300	400

***Tabla 11**: Valores de parámetros significativos de algoritmos genéticos para la ejecución de Grid Search.*

Los valores se establecieron haciendo un análisis de la tendencia de los parámetros de la prueba para el cálculo de ANOVA.

Haciendo uso de la Ecuación 45 de este documento se procede a normalizar los datos.

A partir de la combinatoria de la Tabla 13 se generan 16 Set's de prueba, a continuación se muestra los 5 mejores resultados donde Número de genes a cruza (Ng) y Población Inicial (PbIni):

Parámetro		FO_S		FN_S	
Ng	**PbIni**	**Promedio**	**Desviación**	**Promedio**	**Desviación**
400	**400**	835.06	30.63	0.4731	0.0173
400	**300**	835.92	31.72	0.4736	0.0179
400	**200**	849.80	32.54	0.4815	0.0184
400	**100**	857.78	34.45	0.4860	0.0195
300	**300**	894.52	35.18	0.5068	0.0199

***Tabla 12**: Mejores resultados del Grid Search para el Algoritmo Genético*

Como se observa el número de genes a mutar en el cruce, tiene mayor impacto en la función objetivo a medida que se incrementa el valor, tal como se menciona antes, esto obedece a que el cruce entre genes solo se realiza si existe una mejora en la función objetivo, la población inicial aunque no impacta de igual manera, hace una mejora incrementando el valor pero tiende a estabilizase.

La configuración final del algoritmo queda como se muestra a continuación:

Parámetro	*Valor*
Número de genes a cruzar	400
Población Inicial	400
Probabilidad de Mutación	10
Probabilidad de Cruce	60

***Tabla 13**: Configuración final del Algoritmo Genético.*

4.2.3.4.3 Tendencia del algoritmo

Se realiza un análisis del algoritmo y su comportamiento a medida que se incrementa el número de iteraciones, para este fin se ejecutó las pruebas cada 100 iteraciones hasta llegar a un máximo de 2000. A continuación se muestra la gráfica de tendencia del algoritmo.

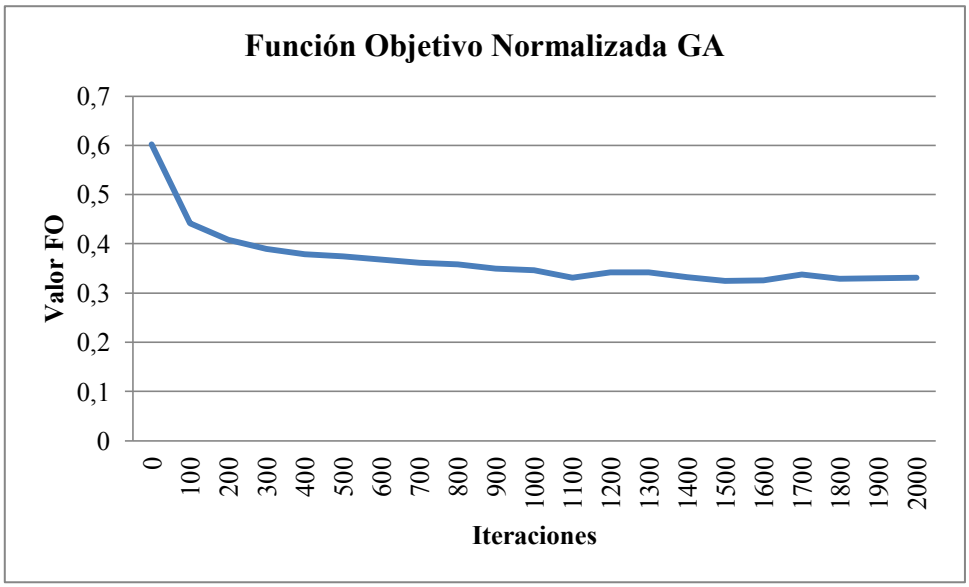


***Ilustración 25**: Tendencia del Algoritmo Genético.*

Entre las iteraciones 0 a 900 hace una mejora constante en la función objetivo de la población inicial estregada por Greedy, luego se estabiliza aunque sigue mejorando. Aunque se hizo un promedio de las iteraciones existen picos que muestra que no siempre hace una mejora del valor anterior.

4.2.4 COLONIA DE HORMIGAS

4.2.4.1 *Colonia de Hormigas MIN – MAX.*

Una de las soluciones propuestas en el presente trabajo tiene como base el algoritmo de Colonia de hormigas con su variante Min – Max (Stutzle & Holger H., 2000), el cual es una mejora sustancial al algoritmo de hormigas planteado por Dorigo y Di Caro (Arito, Leguizamón & Errecalde, 2010).

Por tal motivo, fue necesaria la implementación de esta metodología teniendo en cuenta los siguientes elementos propuestos por Stutzle & Holger:

- Creación de cotas superior e inferior para la cantidad de feromona depositada en la estructura que va a albergarla, de tal manera que el algoritmo no se estanque en óptimos locales y también exista la posibilidad de que aquellas soluciones con poca probabilidad de ser escogidas no se descarten debido a que no poseen feromona para entrar dentro del conjunto solución.
- Tan solo aquella hormiga que logre optimizar la última mejor solución será capaz de actualizar los rastros de feromona.
- Al iniciar o reinicializar los rastros de feromona, estos deben ser creados con el valor equivalente a la cota superior establecida en los parámetros del algoritmo.

- Los rastros de feromona deben reinicializarse cada vez que el algoritmo se encuentre en una situación de estancamiento o en el caso que después de cierto número de iteraciones, no se logre optimizar la última mejor solución hallada por el algoritmo.

4.2.4.2 *Evaluación de la Función Objetivo.*

Inicialmente el algoritmo de colonia de hormigas fue diseñado de tal manera que este realizara una asignación de espacios académicos por fases (primero se asignaba un docente a cada uno de los grupos, después se asignaba un salón y finalmente una franja horaria específica a todos los grupos), pero al realizar una medición del algoritmo, este no obtenía unos buenos resultados comparándolo con los algoritmos de Búsqueda Tabú y Algoritmo Genéticos, ya que el valor de la función objetivos del algoritmo desarrollado por fases obtenía un resultado de función objetivo que se encontraba en los valores de 950. Aunque en cierta manera este valor era considerablemente mejor al arrojado por la asignación manual de la universidad que tenía un valor de 1765 no era fácil realizar una comparativa para este trabajo en cuanto a los datos obtenidos ya que los otros dos algoritmos propuestos realizaban una asignación total a cada uno de los espacios académicos. Es decir se tomaba un primer grupo y se le asignaba un docente, un salón y una franja, en seguida se tomaba otro grupo de asignatura y se realizaba el mismo procedimiento y así sucesivamente hasta asignar todos los recursos a los espacios académicos. Por tal motivo y para efectos comparativos se optó por realizar una asignación de los espacios académicos de la misma manera que lo realizan los otros dos algoritmos propuestos en este documento.

En cuanto a la evaluación de la función objetivo, el algoritmo de colonia de hormigas utiliza la misma metodología adoptada por búsqueda tabú la cual mide restricciones individuales, restricciones por segmento y restricciones a nivel de la solución, de tal manera que al realizar un cambio en algún grupo prometedor no se realiza una medición de la función objetivo a la totalidad de los espacios académicos sino que se mide el impacto de realizar dicho movimiento.

4.2.4.3 *Estructuras utilizadas*

Con el fin de dar una solución óptima a la problemática planteada, fue necesaria la creación de ciertas estructuras, en su mayoría listas java, que fueron las encargadas de albergar los datos en memoria (RAM) para su procesamiento por parte del algoritmo.

La estructura principal dentro del algoritmo es aquella que alberga la feromona de los espacios académicos, esta hace referencia a una lista de números enteros la cual tiene una correspondencia 1 a 1 con todos los espacios académicos existentes para la búsqueda de la solución. Esta estructura alberga un número correspondiente a la

cantidad de feromona que tiene aquel espacio académico al que corresponde su índice, de tal manera que la feromona albergada en el índice 4 hace referencia a la feromona para el espacio académico número 4 y así sucesivamente (Ver Ilustración 26).

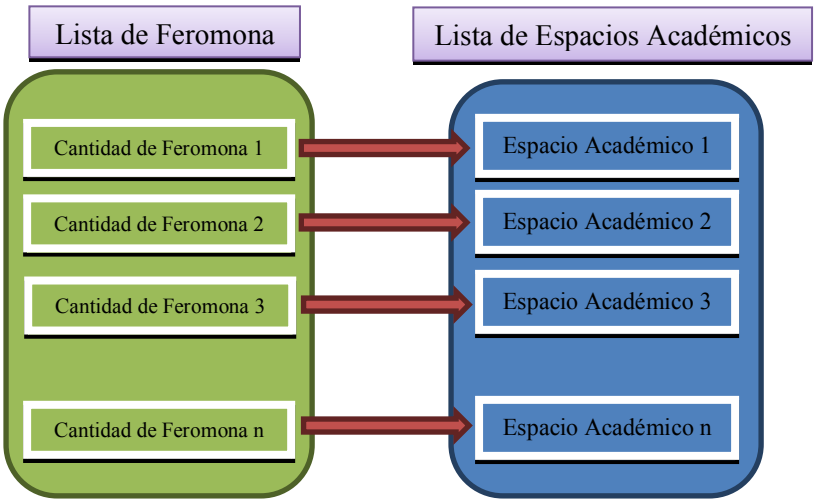


***Ilustración 26**: Feromona Asociada a la Lista de Espacios Académicos*

4.2.4.4 *Metodología de Asignación*

Al igual que Búsqueda Tabú y Algoritmos Genéticos, el algoritmo de Colonia de Hormigas busca optimizar una solución entregada por Greedy. Inicialmente el parámetro principal con el que cuenta el algoritmo es el número de hormigas H, las cuales van a escoger por medio de la metodología ruleta (Explicado en la sección 3.3.4.1.1.2) un grupo candidato para realizar una mejora sobre los recursos asignados a dicho grupo, de tal manera que si el parámetro hormigas se establece como 10, existirán 10 grupos a los cuales se les realizara una modificación sobre los recursos asignados a los mismos (Docente, Franja y Salón) y serán grupos candidatos para ser cambiados en la asignación inicial.

Después de escoger los grupos candidatos, a cada uno de estos se le crean los movimientos correspondientes de sus recursos, de tal manera que se intente mejorar la función objetivo de la solución actual tan solo realizando los movimientos de recursos para el grupo escogido por una hormiga, dado el caso que el parámetro de hormigas H se encuentre establecido como 10 se escogerá la mejor de estas que con los movimientos hechos a el grupo escogido optimice la función objetivo actual, si esta hormiga optimiza la función objetivo actual, la nueva solución planteada con el cambio de grupo se convertirá en la mejor solución y el valor de la función objetivo será actualizado por el obtenido con la modificación hecha por la hormiga. Si ninguna de las 10 hormigas encuentran movimientos para los grupos escogidos de tal manera que estos mejoren la función objetivo actual se incrementa el valor de una variable que lleva el conteo del número de iteraciones sin mejora ξ y se volverán a escoger otros 10 grupos candidatos por medio de la metodología ruleta.

Paralelo a esta asignación y búsqueda de candidatos se encuentra la lista de feromona asociada a cada espacio académico (Ilustración 26), la cual va a ser actualizada cada vez que una hormiga mejore la función objetivo de la solución actual, de acuerdo a este planteamiento si una hormiga encuentra una mejor solución, el valor asociado de la feromona con el grupo modificado será actualizado sumándole un valor de α_{inc} y la feromona asociada a los demás grupos también será actualizada pero en este caso será restándole un valor de α_{dec} a su valor asociado de la feromona, estas actualizaciones de feromona se realizan teniendo en cuenta el tope máximo de la feromona β_{max} y el tope mínimo β_{min}.

Si la variable ξ llega al número máximo de iteraciones sin mejora ξ_{max} se reinicializara el valor de la feromona para todos los espacios académicos al valor β_{max}.

La cantidad de iteraciones que el algoritmo cumpla con este proceso estará limitada por un valor de σ o hasta que el valor de la función objetivo de la solución encontrada sea igual a 0.

4.2.4.4.1 Método Ruleta para la selección de candidatos

Una parte muy importante tiene que ver con la metodología utilizada al realizar la selección de los grupos por cada hormiga.

En el caso del algoritmo basado en colonia de hormigas se escogió el método de la ruleta debido a que proporciona las características necesarias propias del algoritmo de hormigas que consisten en una elección del mejor camino basado en la aleatoriedad pero guiado al mismo tiempo por la probabilidad estimada de hallar una mejor solución (feromona) y puede ser vista como una forma de combinar aleatoriamente los conceptos de exploración del espacio solución con el de explotación de los rastros de feromona.

Además, dentro del conjunto solución, las alternativas con mayor rastro de feromona, poseen mayor probabilidad de ser escogidos, inclinando así la búsqueda en la exploración de estos caminos, pero la naturaleza aleatoria del procedimiento permite escoger alternativas con probabilidad baja, obligando al algoritmo a explorar caminos transitados con poca frecuencia (Garces Ruiz, Granada Echeverri, & Gallego, 2005).

4.2.4.5 *Variables Utilizadas y Pseudocódigo*

Con el fin de proporcionar mayor comprensión en cuanto a la implementación del algoritmo de hormigas, a continuación se presenta la Tabla 14 con las variables utilizadas y la Ilustración 27 con el pseudocódigo de la técnica implementada.

Variable	Descripción
X_{mejor}	Mejor solución
X_{actual}	Solución actual
α	Valor actual de la feromona para todos los grupos
α_{inc}	Factor incremento de la feromona
α_{dec}	Factor decremento de la feromona
H	Numero de hormigas
L	Lista de todos los grupos de Espacios académicos
β_{max}	Valor máximo de la feromona
β_{min}	Valor mínimo de la feromona
ξ	Iteraciones sin mejora
ξ_{max}	Máximo número de iteraciones sin mejora
L_G	Lista de Grupos para los que se deben generar candidatos
G	Grupo
L_C	Lista con todos los candidatos obtenidos a partir de L_G
$L_{E(G)}$	Lista de espacios académicos que pertenecen al grupo G
$\lvert L_{E(G)} \rvert$	Número de espacios que pertenecen al grupo G
M_n	Movimiento número n
$E_n(G)$	Espacio número n que pertenece al grupo G
Ci_G	Candidato número i para el grupo G
Fi_M	Función objetivo con las restricciones individuales del movimiento M
$f_i(M, E)$	Cálculo de las restricciones individuales violadas por el movimiento M al reemplazar el espacio E. (Ilustración 8)
Fz_C	Función objetivo con las restricciones de segmento e individuales si se selecciona el Candidato C
Z_G	Segmento al que pertenece el grupo G
$f_z(C, Z)$	Cálculo de las restricciones por segmento violadas al reemplazar el segmento Z por el creado con los movimientos del candidato C. (Ilustración 9)
Fo_C	Función objetivo (con todas las restricciones violadas) si se selecciona el candidato C
$F_{Xactual}$	Función objetivo de la solución actual (todas las restricciones)
F_{Xmejor}	Función objetivo de la mejor solución (todas las restricciones)
$f_o(C, F)$	Cálculo de todas las restricciones violadas al realizar los movimientos del candidato c. (Ilustración 10)
G_C	Grupo para el que se creó el candidato C
A_C	Mejor candidato en L_C
Fo_{Ac}	Función objetivo del mejor candidato (con todas las restricciones)
G_{Ac}	Grupo para el que se generó el mejor candidato
G_{OC}	Grupos diferentes al que se le genero el mejor candidato
$\alpha(G_{Ac})$	Valor actual de la feromona para el grupo con el mejor candidato en L_C
$\alpha(G_{OC})$	Valor actual de la feromona para los grupos diferentes al que tiene el mejor candidato en L_C
σ	Número máximo de iteraciones

Tabla 14: Descripción de variables usadas en el algoritmo de Hormigas

ACO

$X_{mejor} \leftarrow$ solución Inicial
$X_{actual} \leftarrow X_{mejor}$
$\alpha \leftarrow \beta_{max}$
$\xi \leftarrow 0$
Repetir
$L_G \leftarrow$ Obtener por medio de selección Ruleta H grupos
Para cada G en L_G *// obtener* L_C
Desde n = 0 hasta n < $\left|L_{E(G)}\right|$
Crear M_n para reemplazar a $E_n(G)$
Agregar M_n a Ci_G
$Fi_{Mn} \leftarrow f_i(M_n, E_n(G))$
Actualizar Fz_{CiG} con los valores Fi_{Mn}
Agregar Ci_G a L_C
$Fz_{CiG} \leftarrow f_z(Ci_G, Z_G)$
Actualizar Fo_{CiG} con los valores Fz_{CiG}
$Fo_{CiG} \leftarrow f_o(Ci_G, F_{Xactual})$
fin Para
$A_C \leftarrow$ Mejor candidato en L_C
Ejecutar A_C en X_{actual}
$F_{Xactual} \leftarrow Fo_{Ac}$

Si $F_{Xactual} < F_{Xmejor}$
$X_{mejor} \leftarrow X_{actual}$
$F_{Xmejor} \leftarrow F_{Xactual}$
$\xi \leftarrow 0$

Si $\alpha(G_{AC}) + \alpha_{inc} > \beta_{max}$ *//Si el incremento de la feromona sobrepasa el topemáximo*
$\alpha(G_{AC}) \leftarrow \beta_{max//}$
Sino
$\alpha(G_{AC}) \leftarrow \alpha(G_{AC}) + \alpha_{inc}$ *//Incremento de la feromona*

Si $\alpha(G_{OC})\ \alpha_{dec} < \beta_{min}$ *//Si el decremento de la feromona sobrepasa el topemínimo*
$\alpha(G_{OC}) \leftarrow \beta_{min}$
Sino
$\alpha(G_{OC}) \leftarrow \alpha(G_{OC}) - \alpha_{dec}$ *//Decremento de la feromona*

Sino
$\xi \leftarrow \xi+1$

Si $\xi = \xi_{max}$
$X_{actual} \leftarrow X_{mejor}$ *// reinicio de la búsqueda*
$F_{Xactual} \leftarrow F_{Xmejor}$
$\alpha \leftarrow \beta_{max}$ *//Inicialización de la feromona a el tope Máximo*

Hasta alcanzar σ ó $F(X_{actual}) = 0$

Fin ACO

Ilustración 27: *Estructura del algoritmo de Colonia de Hormigas, fuente propia.*

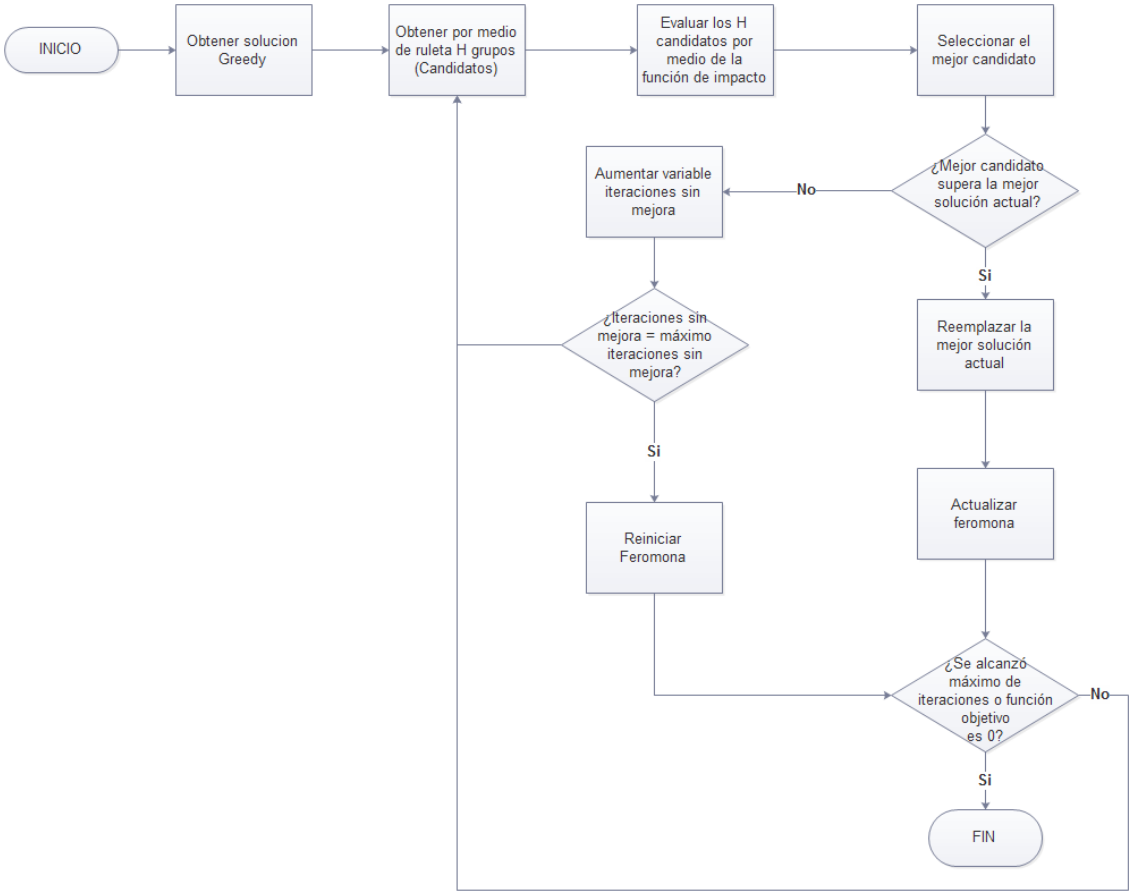


***Ilustración 28**: Diagrama de flujo algoritmo ACO implementado, fuente propia.*

4.2.4.6 *Configuración de Parámetros*

Para realizar un comparativo entre los algoritmos propuestos en este documento se utilizaron los mismos pesos de las restricciones violadas por las soluciones arrojadas del algoritmo que están estipuladas en la Tabla 1.

La búsqueda de variables relevantes por medio del método de *ANOVA* tuvo en cuenta las variables que se encuentran en la Tabla 15.

PARÁMETRO	RESTRICCIÓN
H	Numero de Hormigas
ξ_{max}	Máximo número de iteraciones sin mejora
β_{max}	Valor máximo de la feromona
β_{min}	Valor mínimo de la feromona
α_{inc}	Factor incremento de la feromona
α_{dec}	Factor decremento de la feromona

***Tabla 15**: Parámetros del algoritmo de Colonia de Hormigas*

4.2.4.6.1 Primera Etapa

En las pruebas realizadas para la primera etapa del algoritmo se establecieron 6 *Sets* en los cuales se proponen 5 cambios para cada una de las variables presentes en la Tabla 15, esto quiere decir que se tendrán 30 *sets* de prueba en total. Estos valores van a ser cambiados de uno en uno, es decir, primero se va a cambiar un parámetro con cinco posibles valores y los demás parámetros se dejaran iguales, a continuación se cambiara otra variable con cinco posibles valores y las demás variables no serán modificadas y tendrán los valores del *set* de control o *set Base*. Esta metodología puede ser apreciada de mejor manera en la Tabla 16.

	Base	Numero de Hormigas	Max Iteraciones sin mejora	Max Feromona	Min Feromona	Incremento Feromona	Decremento Feromona
H	10	**20, 40, 60, 80, 100**	10	10	10	10	10
ξ_{max}	20	20	**200, 300, 400, 500, 600**	20	20	20	20
β_{max}	100	100	100	**200, 300, 400, 500, 600**	100	100	100
β_{min}	10	10	10	10	**20, 30, 40, 50, 60**	10	10
α_{inc}	10	10	10	10	10	**30, 50, 60, 80, 100**	10
α_{dec}	10	10	10	10	10	10	**30, 50, 60, 80, 100**

***Tabla 16**: Set's de Prueba para determinar los parámetros significativos de ACO*

Para el cálculo de ANOVA y del Test de Tukey se hizo uso del software MINITAB. Cada uno de los Set's de prueba planteados en la Tabla 16 se ejecutaron 50 veces (250 muestras por parámetro).

En la Ilustración 29 se muestran los datos arrojados al hacer la prueba de los datos ingresados al software de ANOVA.

```
Analysis of Variance

Source    DF    Adj SS   Adj MS  F-Value  P-Value
Factor     5  11902473  2380495   971,75    0,000
Error   1494   3659848     2450
Total   1499  15562321
```

***Ilustración 29**: Tabla ANOVA para ACO. Obtenida con el software MINITAB*

La Ilustración 30 presenta la varianza de cada parámetro con respecto a la función objetivo. En la ilustración se observa que existen dos variables que presentan relevancia en comparación a las demás (Número de Hormigas e Iteraciones sin Mejora), esta grafica fue realizada por el software MINITAB.

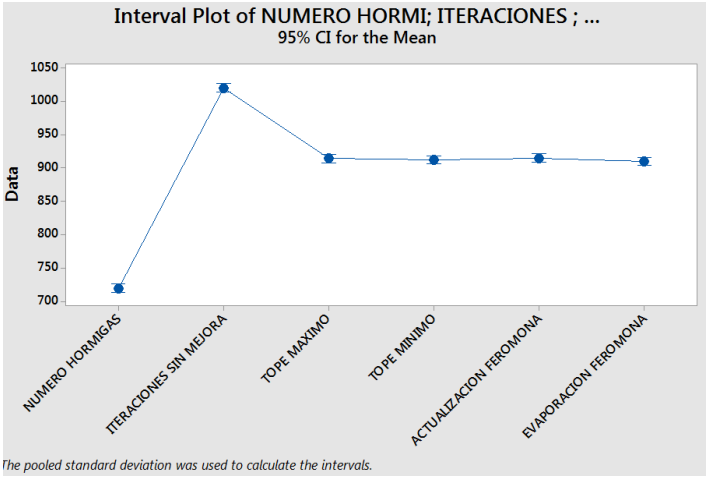


***Ilustración 30**: Varianza de cada parámetro de ACO. Obtenida con el software MINITAB*

La Ilustración 31 muestra el *test de Tukey* que arroja MINITAB, este clasifica las variables de tal manera que aquellas que no compartan una letra son consideradas variables relevantes dentro del algoritmo. En esta ilustración se muestra que las variables relevantes son el Numero de Hormigas H y el número máximo de Iteraciones sin Mejora ξ_{max}.

Ciertamente la cantidad de hormigas que se utilicen para encontrar una mejor solución influye en gran medida a la mejora de la solución de la asignación, debido a que a mayor cantidad de hormigas existen mayor número de posibles cambios en los grupos que mejoren significativamente la función objetivo.

En cuanto a la segunda variable que fue considerada por el cálculo ANOVA, puede deducirse que el algoritmo se ve guiado por la feromona depositada y de alguna manera se queda estancado en óptimos locales, con lo cual la variable correspondiente al número de iteraciones sin mejora cumple a cabalidad su principal función que es la de guiar al algoritmo por nuevos caminos después de que este se quede estancado en un óptimo local, esta es muy seguramente, una razón por la cual el análisis realizado arroja que esta variable resulta ser relevante.

Tukey Pairwise Comparisons

Grouping Information Using the Tukey Method and 95% Confidence

Factor	N	Mean	Grouping
ITERACIONES SIN MEJORA	250	1020,32	A
ACTUALIZACION FEROMONA	250	915,16	B
TOPE MAXIMO	250	914,19	B
TOPE MINIMO	250	912,29	B
EVAPORACION FEROMONA	250	910,25	B
NUMERO HORMIGAS	250	719,84	C

Means that do not share a letter are significantly different.

***Ilustración 31**: Test de Tukey para ACO. Obtenida con el software MINITAB*

4.2.4.6.2 Segunda Etapa

A partir de los resultados obtenidos en la primera etapa de pruebas del algoritmo se utilizó la metodología de *grid search* para determinar aquellos valores que optimizan la solución. Estas pruebas, al igual que las primeras fueron realizadas ejecutando cada *set* de pruebas 50 veces y con ayuda de la herramienta "Perfidix".

	Valor 1	Valor 2	Valor 3	Valor 4	Valor 5
H	20	50	100	150	200
ξ_{max}	20	50	80	100	150

***Tabla 17**: Valores propuestos de las variables relevantes para realizar Grid seach*

Los valores expuestos en la Tabla 17 muestran aquellos datos con los que se hizo la búsqueda *grid seach* para encontrar aquellos con los que el algoritmo de Colonia de Hormigas optimiza la función objetivo para el problema propuesto, estos valores fueron escogidos de manera arbitraria. En total se utilizaron 25 *set's* se prueba (combinatoria de los datos entre las variables relevantes).

En la Tabla 18 se muestra el promedio y la desviación estándar de los 5 mejores *set's* de prueba (FO_s) junto con la normalización de los mismos datos (FN_s).

Parámetro		FO_S		FN_S	
H	ξ_{max}	Promedio	Desviación	Promedio	Desviación
200	50	620.08	28.84	0.3513	0.0163
200	100	620.86	33.04	0.3518	0.0187
200	20	625.88	39.79	0.3546	0.0225
150	150	627.38	27.29	0.3555	0.0155
200	80	628.44	32.87	0.3561	0.0186

***Tabla 18**: Mejores resultados del Grid Search para el algoritmo ACO*

Como se puede observar en los resultados obtenidos, los valores con los que se optimiza la función objetivo son aquellos en los que el número de hormigas es el mayor, en este caso 4 de los 5 parámetros con los que se obtuvo mejor valor fueron los de mayor número de hormigas propuestas. En cuanto al número máximo de iteraciones sin mejora los resultados no son tan dicientes debido a que en los cinco mejores resultados se obtuvo un número diferente en el mismo.

El primer Set en la Tabla 18 presenta el mejor promedio y la segunda desviación estándar más pequeña, por tal razón la configuración del algoritmo ACO que será usada en las comparaciones será la siguiente:

Parámetro	Valor
H	200
ξ_{max}	50
β_{max}	100
β_{min}	10
α_{inc}	10
α_{dec}	10

Tabla 19: *Configuración final del algoritmo ACO*

4.2.4.6.3 Tendencia del algoritmo

En la Ilustración 32 se puede observar la tendencia del algoritmo de Colonia de Hormigas en cuanto al número de iteraciones y su función objetivo normalizada. Esta grafica fue realizada haciendo una medición del algoritmo cada 100 iteraciones hasta llegar a 2000, hay que tener en cuenta que para hallar cada valor mostrado en la gráfica se obtuvo el promedio de 50 corridas con cada número de iteraciones.

A simple vista se puede apreciar una mejora súbita en cuanto a la solución entregada por el algoritmo *Greedy* (0 iteraciones) y las primeras 100 iteraciones del algoritmo de Colonia de hormigas, esto quiere decir que desde muy temprano, el algoritmo cumple a cabalidad su propósito de entregar una mejor solución.

Adicionalmente, al avanzar en el eje de las iteraciones (Eje X) se puede observar una leve mejoría a medida que se va aumentando el número de iteraciones con las que se corre el algoritmo, con lo cual se puede concluir además que a mayor número de iteraciones el algoritmo reduce la función objetivo.

Finalmente parece haber una asíntota horizontal que se encuentra muy próxima al valor 0.3, con lo cual podríamos deducir que muy seguramente sin importar que valor ingresemos a los parámetros de entrada del algoritmo, la solución no va a estar muy lejana del valor normalizado de 0.3.

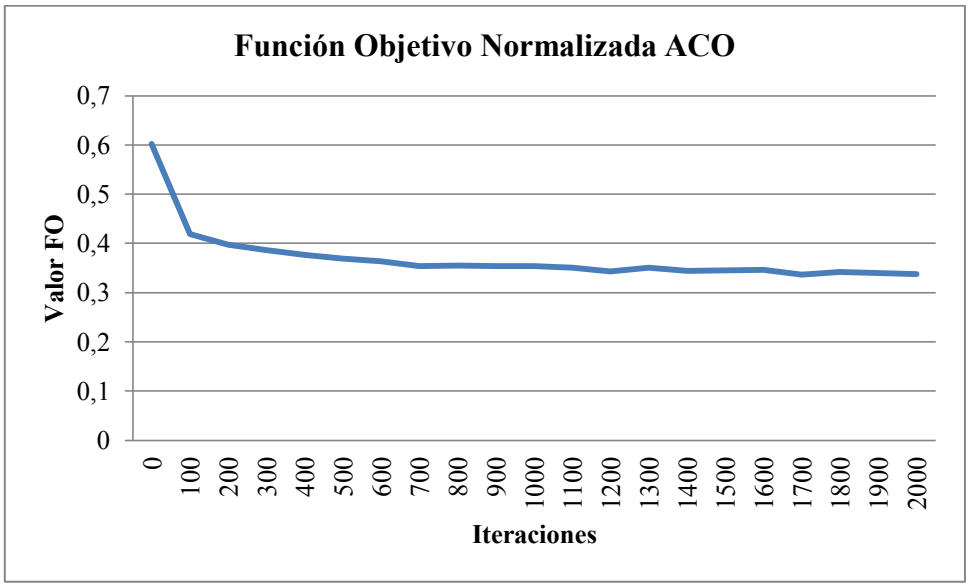


***Ilustración 32**: Tendencia del algoritmo ACO*

4.2.5 RESULTADOS OBTENIDOS.

Antes de presentar los resultados obtenidos, la Tabla 20 compara las metodologías y los conceptos aplicados en las diferentes técnicas. Tal y como se puede observar, la mayoría de los conceptos son aplicados en los tres algoritmos implementados, a excepción de la perturbación y la mutación, utilizados en ATS y GA respectivamente.

ATS	GA	ACO
Profundidad	Población Inicial	Número de Hormigas
Candidato	Gen	Candidato
Solución	Cromosoma	Solución
Iteraciones sin mejora	N/A	Iteraciones sin mejora
Creación de movimientos	Cruce	Creación de movimientos
Selección aleatoria del grupo para el que se generan candidatos	Selección por torneo de dos cromosomas padres para hacer el cruce	Selección probabilística guiada por la Feromona

Tabla 20: *Comparativa de aspectos comunes en cada algoritmo*

Adicionalmente, es necesario resaltar uno de los aspectos más sobresalientes de la implementación realizada, la cantidad de restricciones. Pese al gran número de trabajos relacionados en la literatura, ninguno contempla la totalidad de las variables presentadas en este desarrollo. La Tabla 21 compara los trabajos más sobresalientes frente a las restricciones consideradas por el modelo abordado en el presente documento; la letra P indica que se realizó alguna configuración previa o pre asignación por lo que la restricción no fue contemplada como tal.

Trabajo	Objetivo	Dco	Dsc	Dfd	Dfp	Dsa	Sco	Sfd	Gsf	Ghi	Gti	Gnf	ssc	sri	gsd	gjc	gdc	gel	gss	dci
(Martínez Ruiz, Sánchez García, Muñoz Arteaga, & Castañeda Ramírez, 2006)	Asignar las asignaturas que el docente debe dictar teniendo predefinida la dupla asignatura salón.	X		X	X	X	X	P		X	X				X				X	
(Soria Alcaraz, Carpio, & Puga, 2010)	Asignar los horarios de los estudiantes minimizando los conflictos.	X			X	X	X		X	X					X				X	
(Pérez de la Cruz & Ramírez Rodríguez, 2011)	Busca asignaciones sin cruces y eventos a cierta distancia de tiempo.						X		X									X	X	
(Socha, 2003)	Generación de horarios para estudiantes. Busca que las características requeridas por una asignatura sean satisfechas por el salón asignado.						X		X	X			X	X					X	
(Ayob & Jaradat, 2009)	Plantean 2 versiones del algoritmo propuesto por (Socha, 2003), por lo que tienen las mismas variables.						X		X	X			X	X					X	
(Zhang & Lau, 2005)	Realizar asignaciones sin cruces, teniendo en cuenta que el salón tenga la capacidad para albergar a los estudiantes.						X		X	X			X						X	
(Castro Medina & Medaglia, 2004)	Busca maximizar el uso de los salones a través de la asignación única de los slots (salón - franja horaria), ya que las secciones (docente - grupo de estudiantes) están previamente definidas (no se asignan docentes).	P			P	P	X	X	X	P					P				X	P
(Astaiza A., 2005)	Programación de exámenes que contempla asignaturas, estudiantes y bloques de tiempo.						X		X	X			X						X	
(Mendez Giraldo, Caballero Villalobos, & Álvarez Pomar, 2007)	Busca generar el horario de los 3 semestres que dura el postgrado, garantizando que en ese periodo de tiempo, todas las asignaturas sean vistas por todos los estudiantes								X	X						X	X	X		
(Peñuela, Franco B., & Toro O., 2008)	Asignar salones que cumplan con los requerimientos del evento (conjunto de características requeridas). Los estudiantes se encuentran previamente inscritos						X	X	X				X	X					X	

Tabla 21: *Aspectos no contemplados en trabajos relevantes*

4.2.5.1 *Datos de Prueba*

Los datos utilizados para las pruebas fueron los proporcionados por el proyecto curricular de Ingeniería de Sistemas para el semestre 2012-III. Fue necesario realizar ajustes en los datos recibidos debido a que la información dada por el proyecto se encontraba con algunas inconsistencias como falta de información o conflictos entre algunas asignaciones[7].

Recurso Académico	Cantidad
Docentes	80
Salónes	64
Asignaturas	76
Grupos	184
Franjas	36
Tipos de Salón	5
Recursos	3

Tabla 22: *Cantidad de datos utilizados en las pruebas realizadas. Datos del semestre 2012-III*

4.2.5.2 *Comparativa entre Algoritmos*

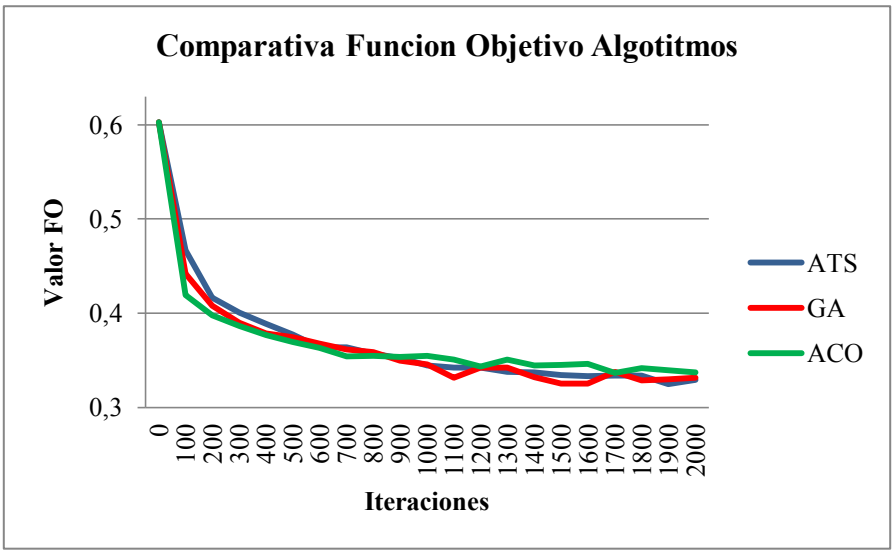


Ilustración 33*: Comparativo de rendimiento de Algoritmos, iteraciones Vs Función Objetivo normalizada*

[7] La información proporcionada por el proyecto curricular no se encontraba actualizada en el momento en que fue suministrada para el desarrollo de este documento. Por esta razón se presentaban las inconsistencias.

La Ilustración 33 presenta la comparativa de los tres algoritmos en términos de la función objetivo. Los picos que se observan en la gráfica son ocasionados por la variabilidad de los algoritmos, pese a que cada prueba se ejecutó 50 veces.

En términos de variabilidad, el algoritmo más estable es ATS ya que presenta el menor número de picos, pero resulta ser el que posee la mayor función objetivo hasta la iteración 500, es decir, es la implementación que más tarda en mejorar la asignación realizada por el algoritmo Greedy. Por otra parte, ACO mejora rápidamente la asignación inicial pero termina con los valores más altos de la función objetivo.

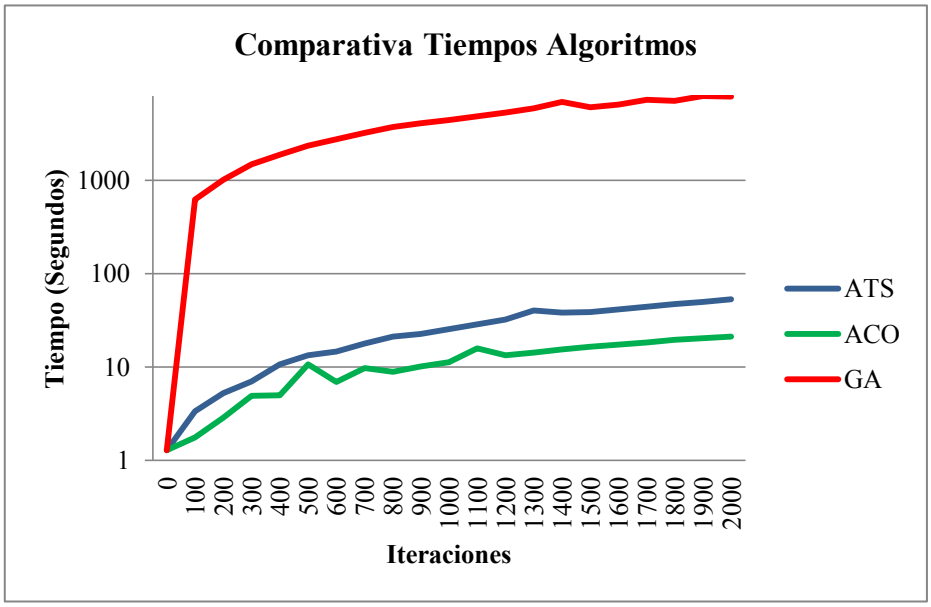
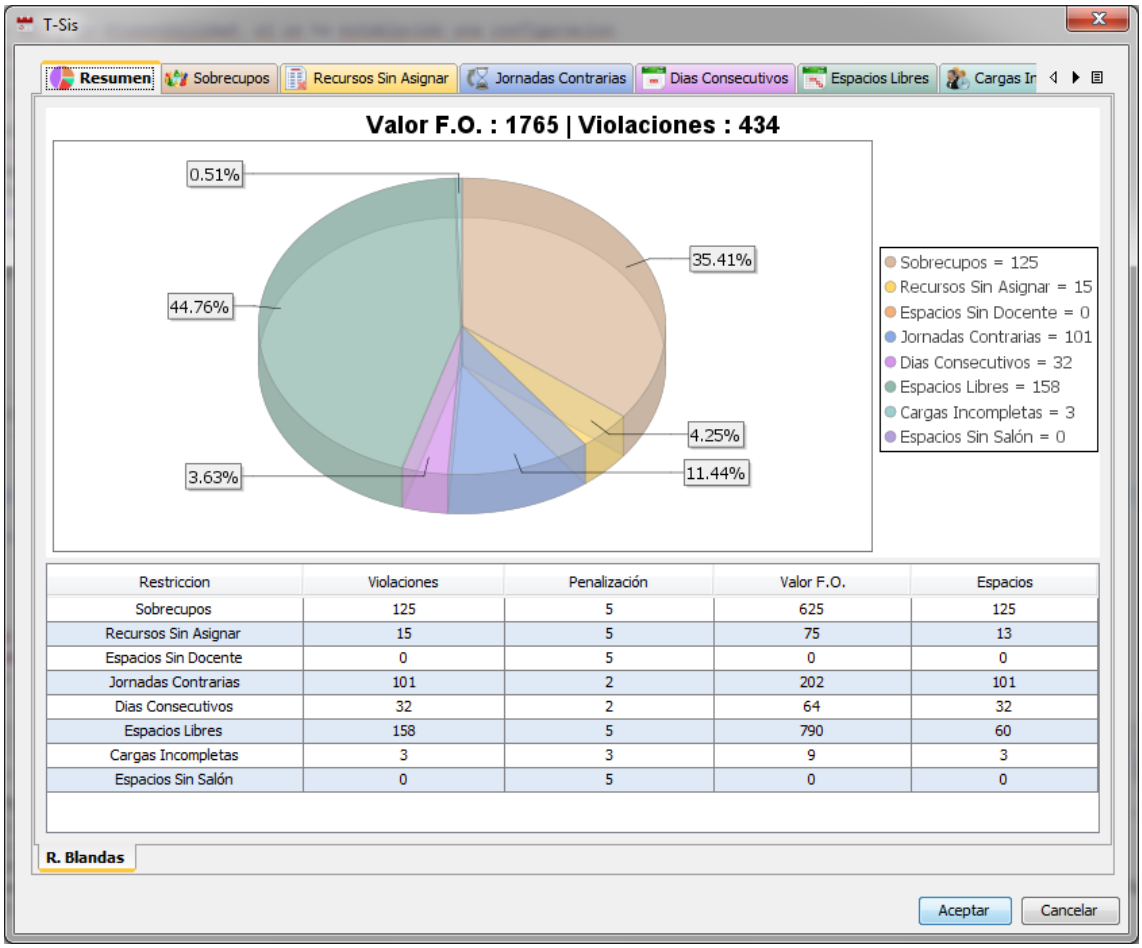


***Ilustración 34**: Comparativa de tiempos de ejecución de algoritmos, iteraciones Vs tiempo (Segundos)*

Para analizar el rendimiento de cada una de las implementaciones realizadas, se presenta la Ilustración 34 con la comparativa de las técnicas en términos de tiempo. Como se puede observar GA es el algoritmo que requiere de mayor tiempo para entregar una solución, mientras que ATS y ACO son similares en este aspecto. La diferencia con el algoritmo genético se debe a que al inicio se genera una población inicial donde se ejecuta cierto número de veces el algoritmo Greedy, y en cada iteración se intenta hacer una mejora a las soluciones seleccionadas (padres), de modo que GA trabaja con 400 soluciones (determinadas por el parámetro población) mientras que ATS y ACO trabajan solo 2. El pico que se observa en GA corresponde a la generación de la población inicial.

4.2.5.3 *Comparativa Asignación Actual Vs Asignación del Aplicativo*



Restriccion	Violaciones	Penalización	Valor F.O.	Espacios
Sobrecupos	125	5	625	125
Recursos Sin Asignar	15	5	75	13
Espacios Sin Docente	0	5	0	0
Jornadas Contrarias	101	2	202	101
Dias Consecutivos	32	2	64	32
Espacios Libres	158	5	790	60
Cargas Incompletas	3	3	9	3
Espacios Sin Salón	0	5	0	0

Ilustración 35: *Resultado de la asignación de espacios académicos hecha por el proyecto curricular en el semestre 2012-III. Imagen del Software T-Sis*

La ilustración anterior presenta el resumen de las restricciones violadas por la asignación manual, realizada con la información recibida por parte del proyecto curricular. Dicha asignación presenta un total de 434 restricciones violadas y un valor de 1765 en la función objetivo, valor con el que se normalizaron los resultados de los algoritmos. La Ilustración 36 compara esta asignación contra las tres técnicas implementadas.

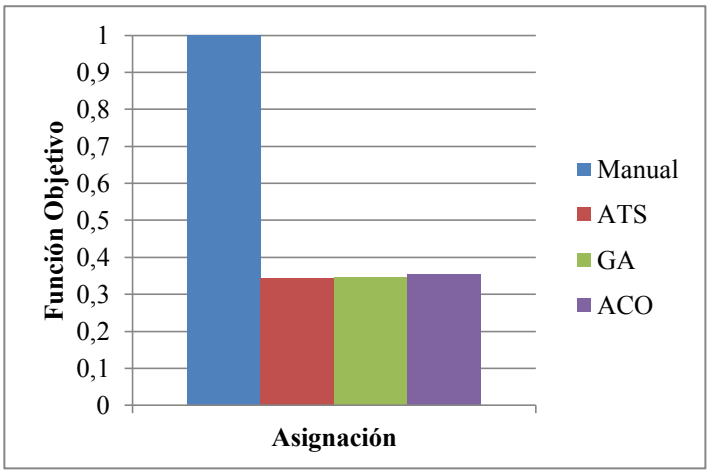


Ilustración 36: *Comparativa Asignación Manual vs Algoritmos implementados*

La ilustración anterior permite evidenciar que todos los algoritmos producen mejoras significativas sobre la asignación manual. Las tres técnicas reducen en más de un 50% el valor de la función objetivo, y teniendo en cuenta los tiempos de ejecución de las técnicas, resultaría beneficioso el uso del prototipo desarrollado para la generación de horarios.

Aunque la ejecución de GA tarda unos pocos minutos en generar una solución, con todos los algoritmos se consigue una mejora sustancial en el tiempo requerido para generar el horario de un semestre, con respecto al proceso actual. Con la asignación manual se requiere de varios días para la creación del horario, y son pocos los cambios que se pueden realizar después de que este ha sido creado. Al implementar el desarrollo abordado en el presente documento, el tiempo que antes se invertía para generar un horario puede ser utilizado para generar múltiples soluciones y generar las modificaciones a las que haya lugar. El prototipo entregado agrega flexibilidad al proceso de asignación al facilitar los cambios a un horario generado, tal y como se muestra a continuación.

4.2.6 PRESENTACIÓN FINAL DEL APLICATIVO.

Se desarrolló un software prototipo denominado T-Sis (acrónimo de *Timetabling System*) con el que se ofrecen diferentes funcionalidades que permiten una mayor flexibilidad en el proceso de asignación, con respecto a la metodología actual. Aunque en el manual de usuario se describe a cabalidad el manejo de la aplicación, a continuación se presentan las características más sobresalientes en el prototipo desarrollado:

1. Consultar el horario semanal de asignaturas, docentes o salones.
2. Imprimir cada una de las consultas
3. Visualización de las restricciones suaves violadas.
4. Edición manual del horario generado[8].
5. Generación de 2 o más horarios[9].

4.2.6.1 *Generación de Varios Horarios*

La aplicación permite al usuario generar varios horarios y seleccionar qué tipo de técnica desea implementar en cada uno de ellos, tal y como se presenta en la Ilustración 37. Es necesario resaltar que los parámetros con los que se ejecuta cada algoritmo son los valores obtenidos en las pruebas realizadas (Tabla 7, Tabla 13 y Tabla 19).

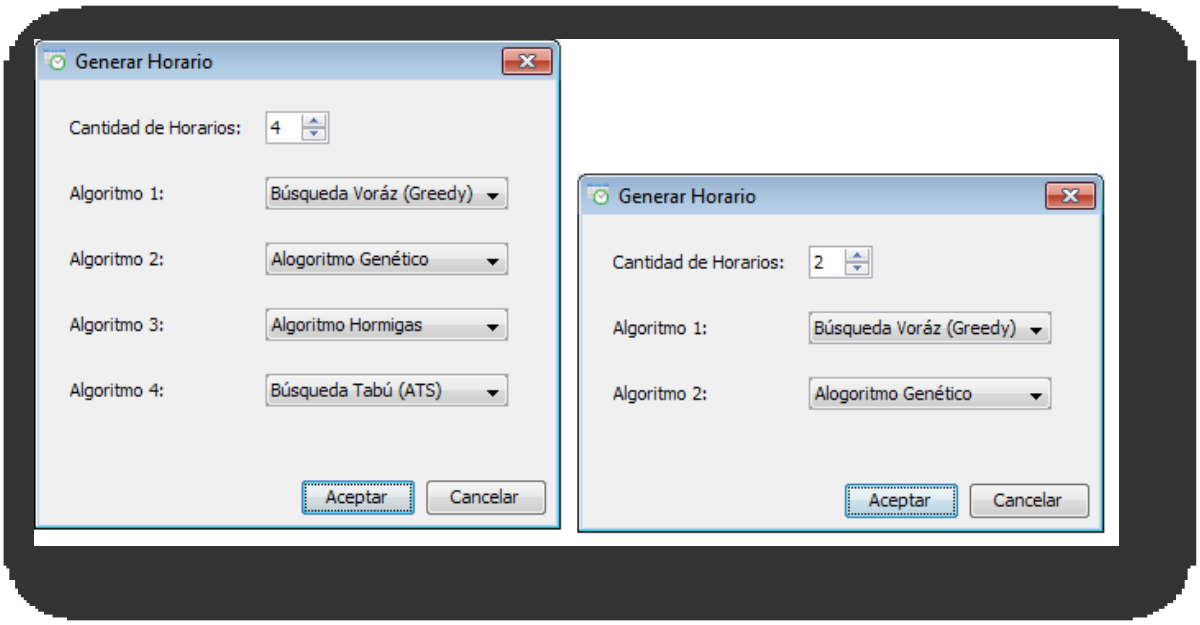


Ilustración 37: *Ventana para la generación de múltiples horarios*

Una vez el usuario configura el número de horarios y las técnicas a utilizar, el aplicativo ejecuta el proceso de asignación y presenta los resultados en la siguiente tabla:

[8] El sistema verifica si el cambio a realizar es posible o no. En caso negativo, el cambio no tendrá lugar y se mantendrá el horario generado.
[9] El sistema presenta 2 o más soluciones, y aunque se muestra la cantidad de restricciones suaves violadas en cada una de ellas, será el usuario quien defina cual aceptar.

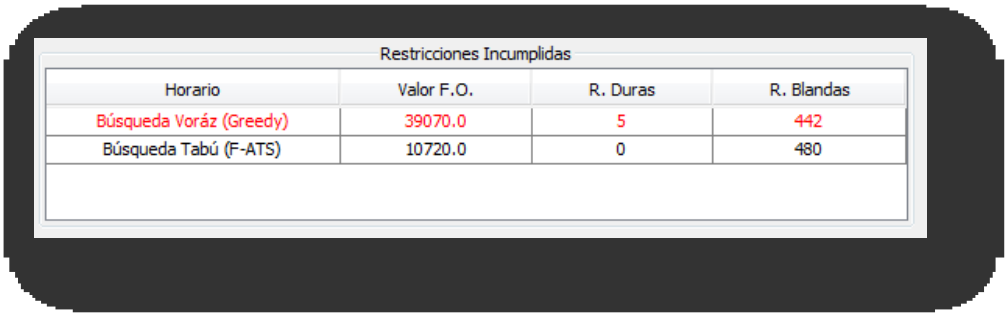


Ilustración 38: *Tabla que presenta los horarios generados en la aplicación*

En dicha tabla se presenta la información relevante que permite identificar la calidad del horario generado. Se muestra el valor de la función objetivo, el número de restricciones duras, el número de restricciones blandas y el algoritmo generador de la asignación. Si una asignación presenta restricciones duras violadas, se considera como una solución infactible y se muestra en rojo en la tabla de restricciones, tal y como se observa en la Ilustración 58.

4.2.6.2 *Visualización de restricciones violadas*

El usuario puede seleccionar uno de los horarios generados y ver en detalle las restricciones violadas.

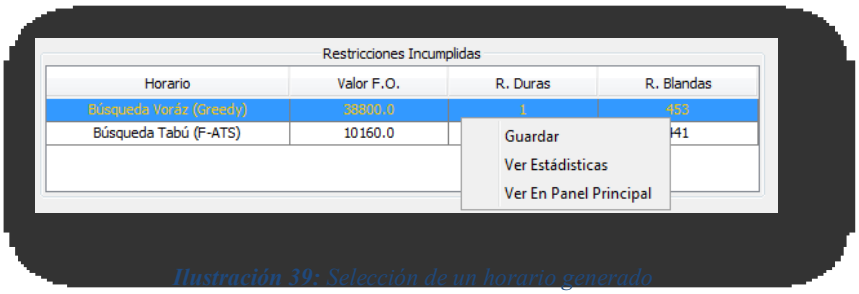
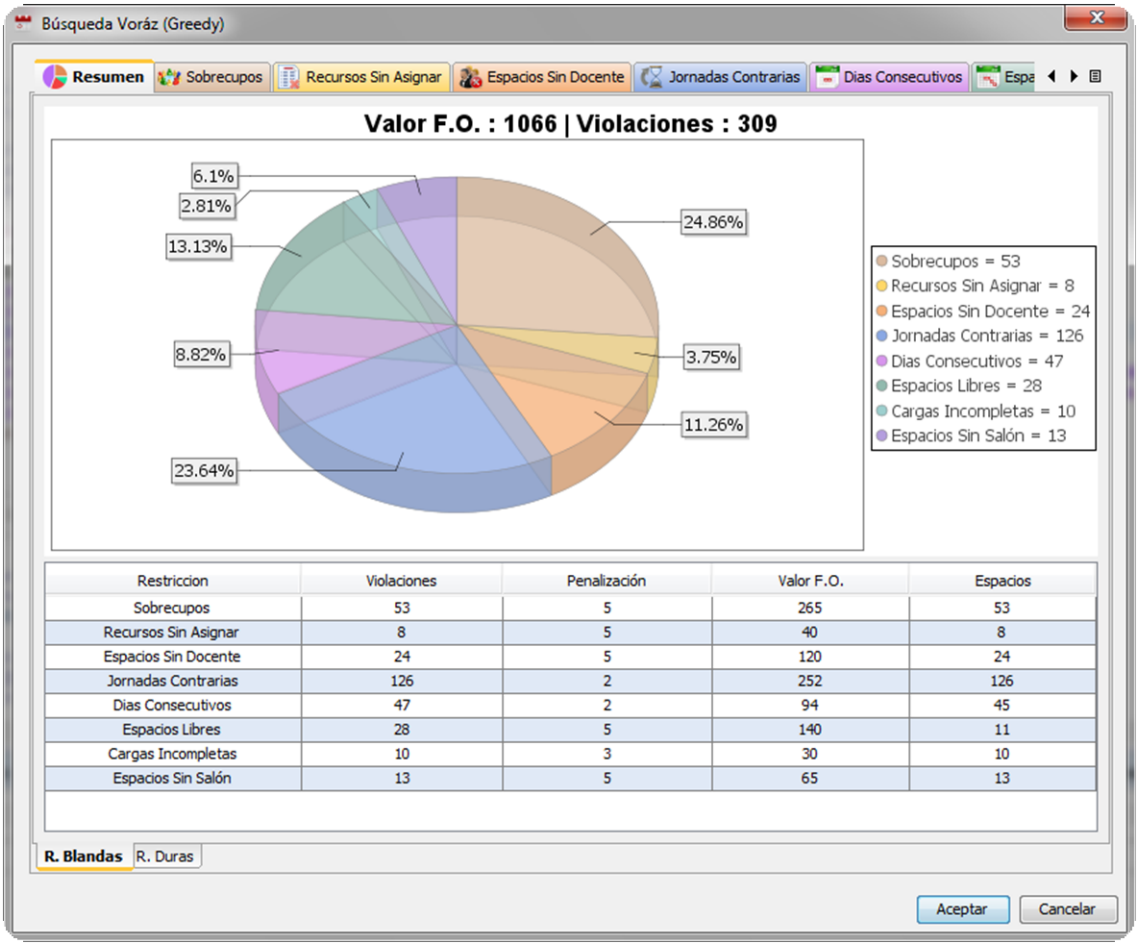


Ilustración 39: Selección de un horario generado

Al acceder a la opción de ver estadísticas se presenta una ventana con los detalles de la asignación.



Restriccion	Violaciones	Penalización	Valor F.O.	Espacios
Sobrecupos	53	5	265	53
Recursos Sin Asignar	8	5	40	8
Espacios Sin Docente	24	5	120	24
Jornadas Contrarias	126	2	252	126
Dias Consecutivos	47	2	94	45
Espacios Libres	28	5	140	11
Cargas Incompletas	10	3	30	10
Espacios Sin Salón	13	5	65	13

***Ilustración 40**: Ventana que presenta la información de las restricciones violadas*

En esta ventana se presenta una gráfica con el porcentaje de cada restricción violada sobre el valor total de la función objetivo. Las restricciones duras y blandas se presentan en la parte inferior en pestañas diferentes.

Restriccion	Violaciones	Penalización	Valor F.O.	Espacios
Sobrecupos	53	5	265	53
Recursos Sin Asignar	8	5	40	8
Espacios Sin Docente	24	5	120	24
Jornadas Contrarias	126	2	252	126
Dias Consecutivos	47	2	94	45
Espacios Libres	28	5	140	11
Cargas Incompletas	10	3	30	10
Espacios Sin Salón	13	5	65	13

R. Blandas R. Duras

***Ilustración 41:** Tabla inferior de la ventana con los detalles de las restricciones blandas violadas*

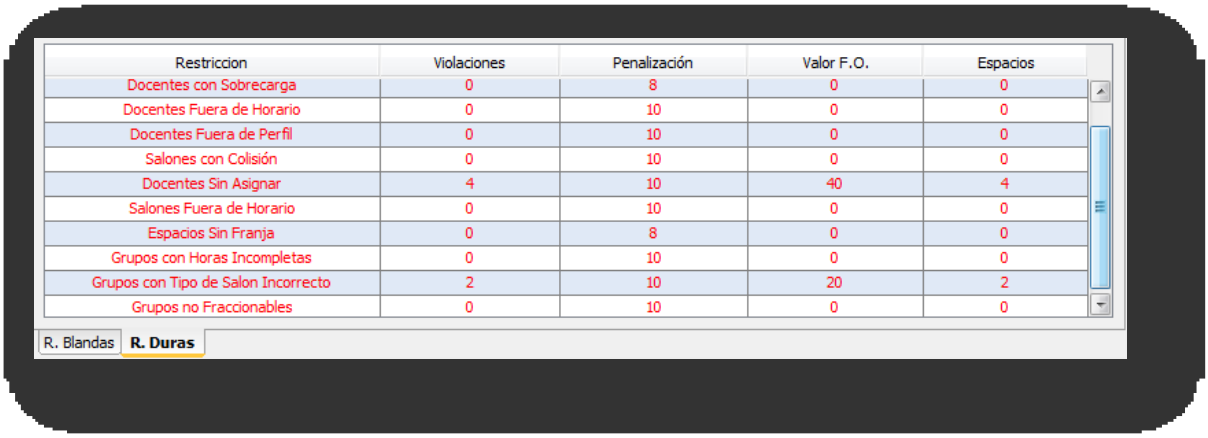


***Ilustración 42**: Tabla inferior con los detalles de las restricciones duras violadas*

Al presentar las restricciones violadas se describe el número de violaciones, la penalización de cada restricción, el valor en la función objetivo obtenido al multiplicar el número de violaciones por la penalización, y la cantidad de espacios que se vieron afectados. En algunas restricciones el número de espacios afectados y la cantidad de violaciones pueden diferir, tal y como sucede con la restricción de recursos sin asignar, donde es posible que un docente requiera varios recursos pero que el salón asignado no posea ninguno de ellos, generando varias violaciones en un único espacio.

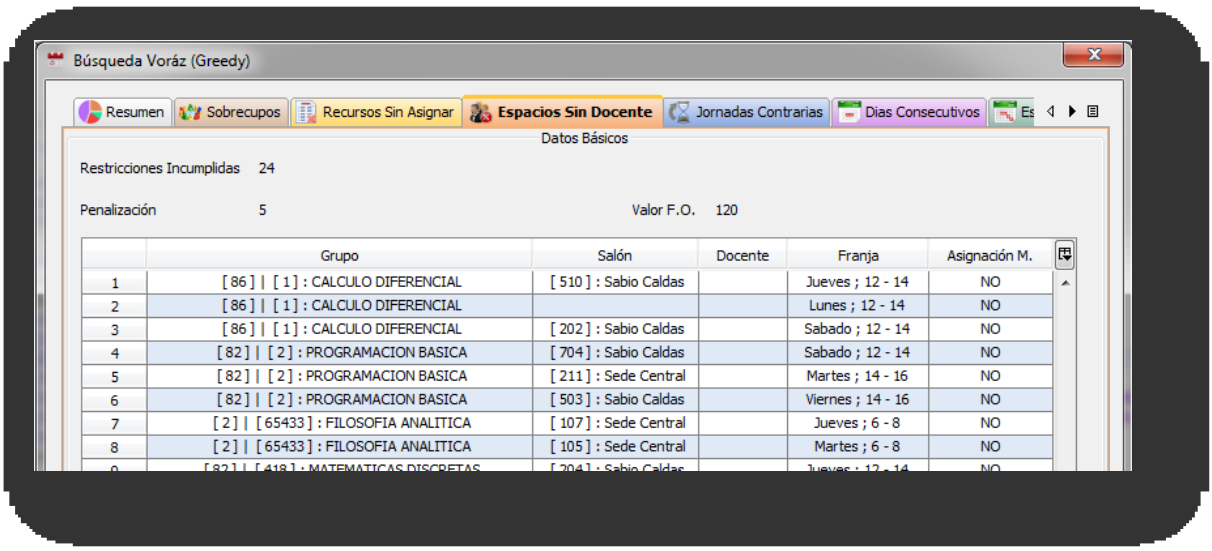


***Ilustración 43:** Detalle de las restricciones violadas*

En cada una de las pestañas se presenta el detalle de las restricciones violadas. La Ilustración 43 presenta la información referente a la violación de la restricción dura "espacio sin docente", en donde se muestran aquellas asignaciones que no poseen docente.

4.2.6.3 *Edición manual del horario generado*

El aplicativo permite que el usuario modifique las asignaciones realizadas, a través de la ventana de que se muestra en la Ilustración 44.

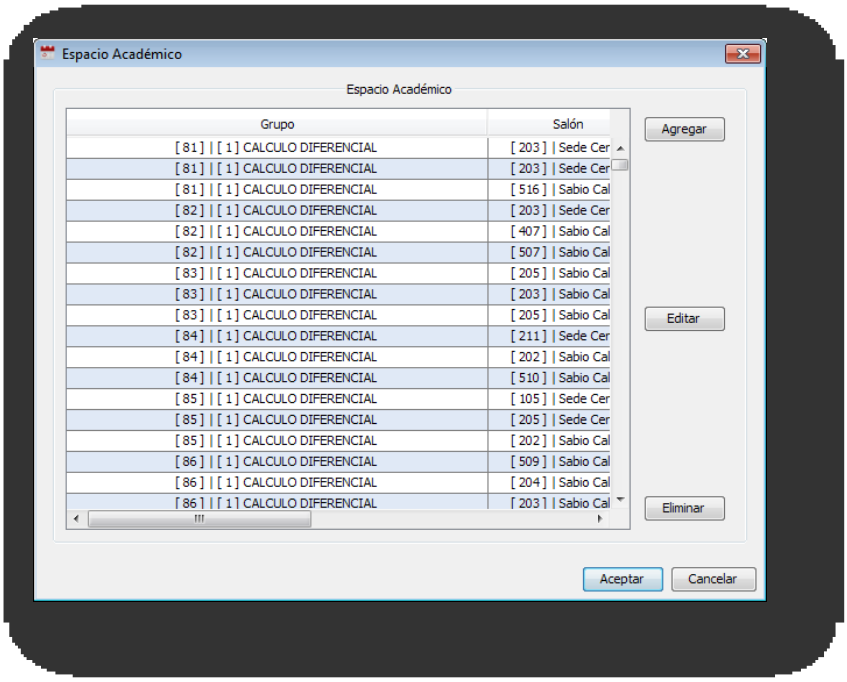


Ilustración 44: *Ventana en la que se listan las diferentes asignaciones realizadas*

Dicha ventana da la posibilidad de agregar, editar o eliminar una asignación.

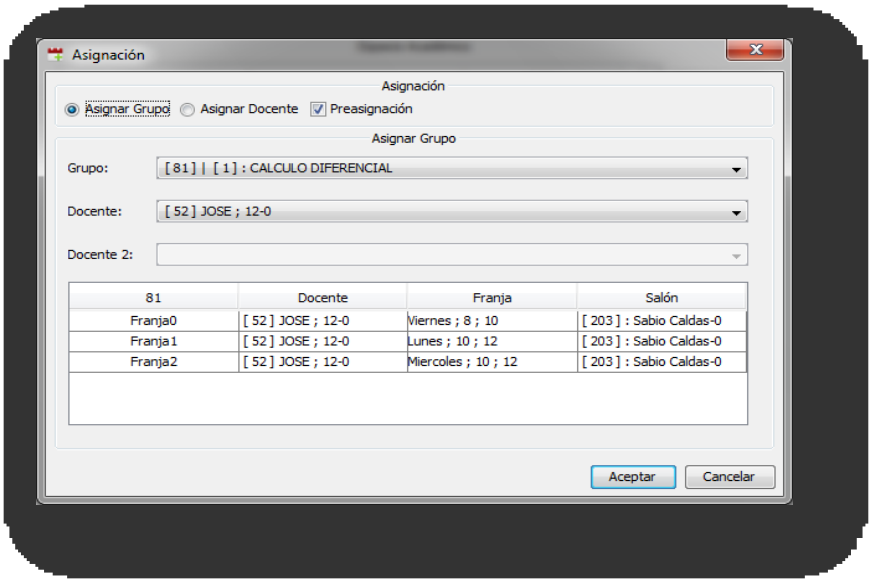


Ilustración 45: *Ventana para realizar una asignación manual*

El usuario selecciona si desea realizar la asignación por grupo o por docente. En el primer caso debe seleccionar primero el grupo y luego se listarán los docentes que pueden ser asignados. Si el usuario escoge realizar la asignación por docente, se presentará la lista de docentes, y con base en el seleccionado, se listarán los grupos que pueden ser dictados por él (para un mayor detalle revisar el manual de usuario).

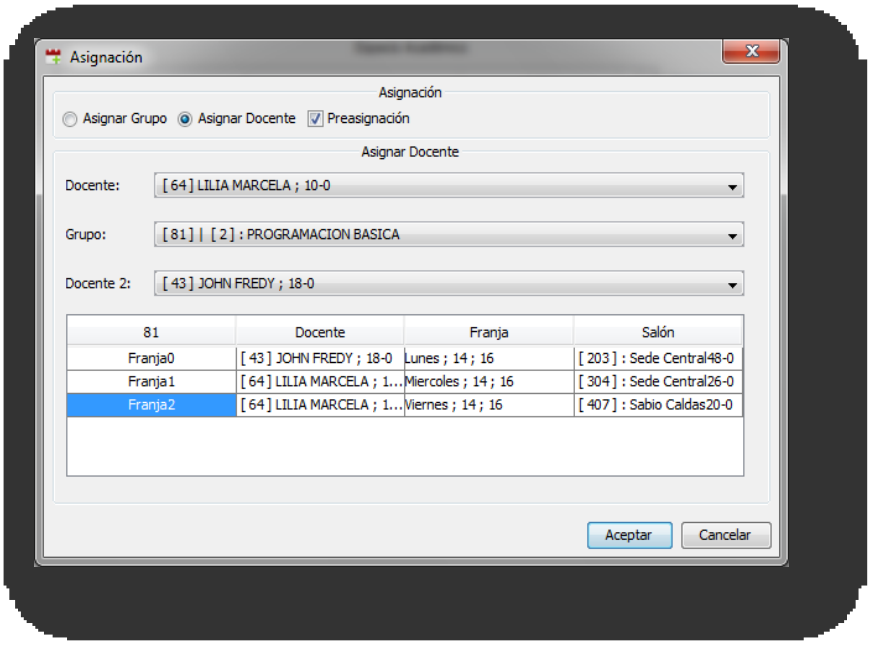


Ilustración 46: *Ventana de asignación manual, ejemplo con 2 docentes*

Si la asignatura a la que pertenece el grupo permite la asignación de 2 docentes, se habilita una lista adicional para seleccionar el segundo docente. Si el usuario desea realizar una asignación antes de generar un horario, puede marcar la opción de pre-asignación obligando a que las diferentes técnicas respeten la asignación manual y esta no se modificada en el proceso automático.

4.2.6.4 *Visualización e Impresión de Horarios*

A través de una ventana de consultas, la aplicación permite al usuario revisar los docentes, salones y grupos que han sido asignados, y consultar posteriormente el horario.

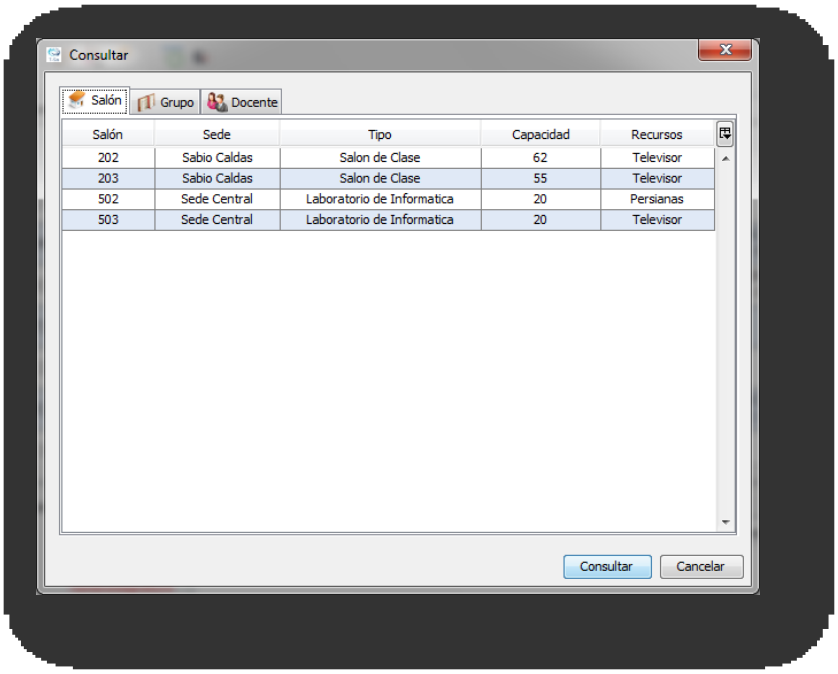


Ilustración 47: *Ventana para consultar los horarios de salón, docente y grupo*

Una vez el usuario selecciona uno de los elementos de la tabla, se presenta el horario tal como se muestra en las Ilustraciones 68, 69 y 70.

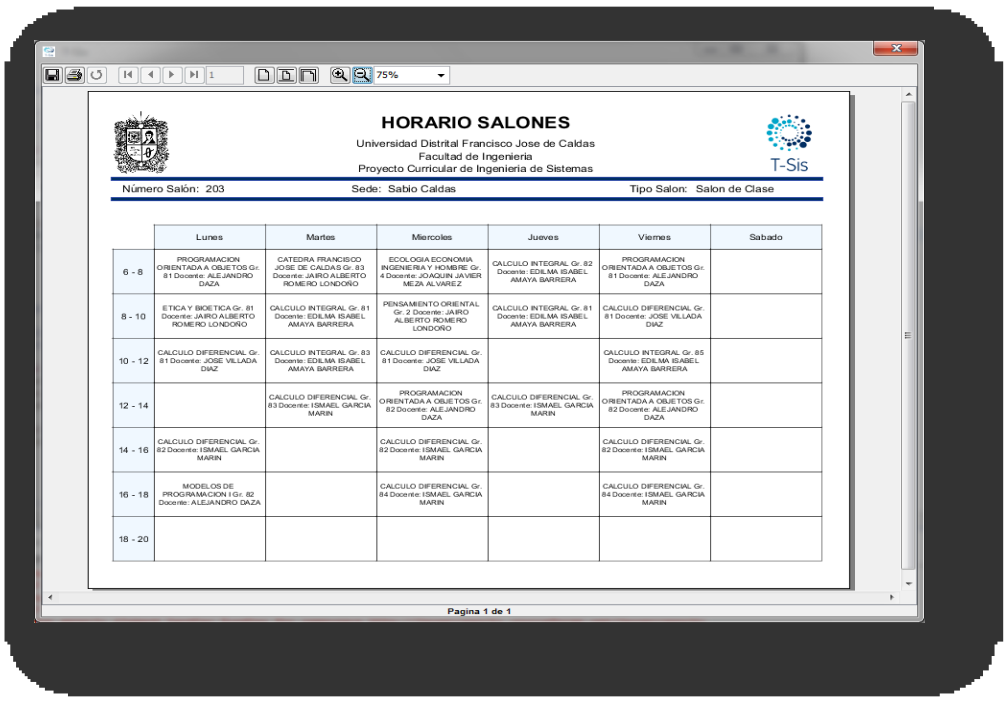


Ilustración 48: *Ejemplo de un horario por salón*

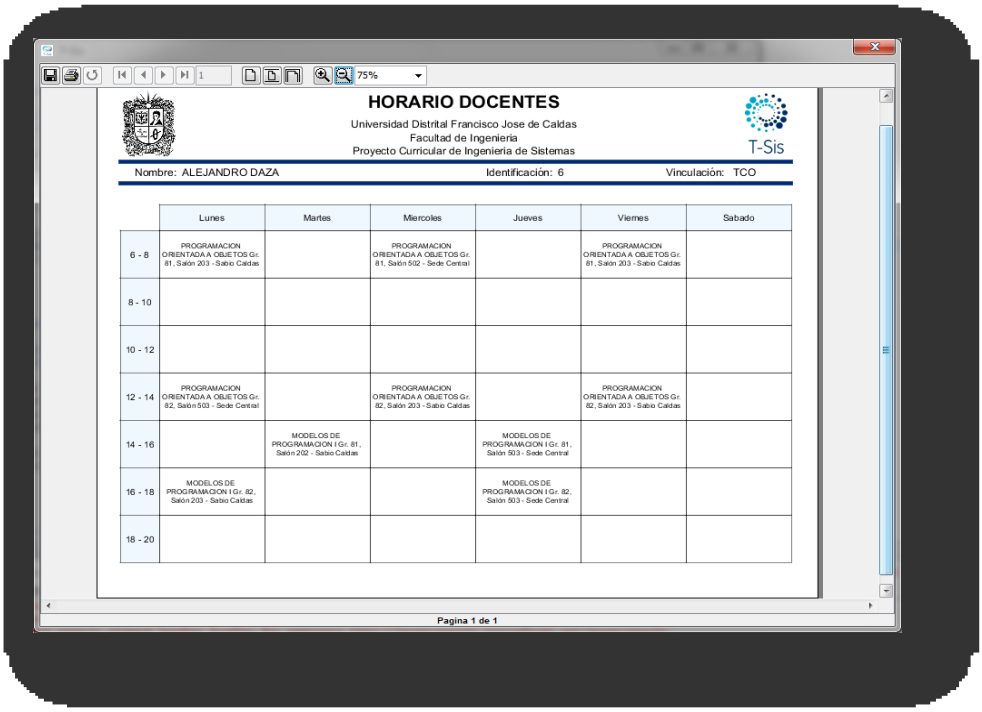


Ilustración 49: *Ejemplo de un horario por docente*

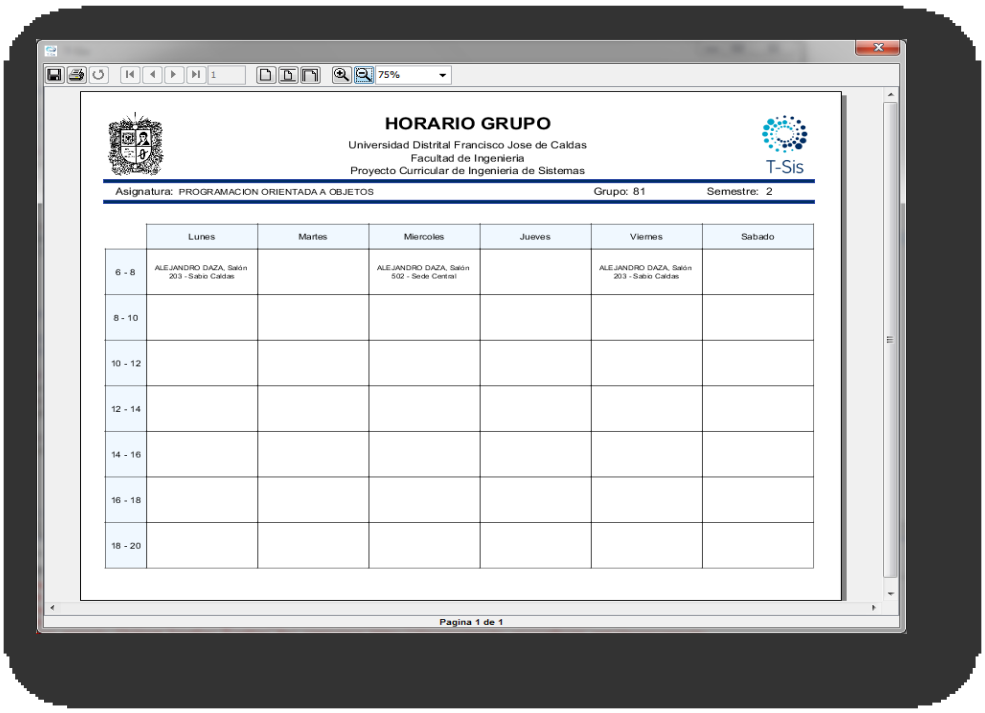


Ilustración 50: *Ejemplo de un horario por grupo*

El horario generado refleja las asignaciones semanales. Las filas representan los bloques de horas o franjas, mientras las columnas los días de la semana. En cada celda se presentan los datos que complementan la asignación: para grupo se mostrará docente y salón asignados, para salón, el docente y el grupo, y para un docente, el grupo y el salón.

5 CONCLUSIONES

Todos los algoritmos desarrollados entregan soluciones que presentan un menor número de violaciones que la asignación manual hecha por la Universidad, y teniendo en cuenta que el modelo implementado considera muchas más variables que las que contemplaría la persona encargada de la creación del horario, el desarrollo resulta beneficioso ya que no solo se provee una herramienta de apoyo que permite crear asignaciones en un tiempo mucho menor al del proceso actual, sino que facilita la creación de horarios con mayor calidad y flexibilidad. Se le otorga al usuario la posibilidad de editar los horarios según sea necesario, generar pre asignaciones y modificaciones a uno ya creado, e incluso generar varios horarios y escoger el que le resulte conveniente.

Adicionalmente, las tres técnicas presentan una curva de mejora similar, en las primeras iteraciones se reduce el valor de la función objetivo de manera significativa, para luego estabilizarse generando pequeñas mejoras. Aunque el algoritmo más eficiente en términos de tiempo es ACO, seguido de cerca por ATS, todos, incluyendo GA, entregan soluciones bastante rápidas y el proceso de asignación que actualmente tarda días se reduce a unos pocos minutos. La alta velocidad con la que se ejecutan los algoritmos surge como una exitosa consecuencia de la medición del impacto en la función objetivo.

De cualquier manera el aplicativo desarrollado representa una buena herramienta de apoyo para la persona encargada de realizar la asignación de Espacios Académicos dentro del proyecto curricular de Ingeniería de Sistemas de la Universidad Distrital Francisco José de Caldas al optimizar tres aspectos diferentes: tiempo, calidad y flexibilidad.

6 TRABAJO FUTURO

Una de las primeras líneas de continuación de este proyecto consiste en la unificación con el sistema Cóndor de la Universidad Distrital Francisco José de Caldas. La aplicación desarrollada se encargaría de proveer los datos necesarios para la asignación de los recursos académicos (Salones, Asignaturas, Grupos y Docentes) y de generar el horario previo al proceso de inscripción de asignaturas por parte de los estudiantes. Un mecanismo que podría permitir la colaboración entre estos sistemas sería la creación de un ETL que envié los datos generados por la aplicación y los cargue en el sistema Cóndor, teniendo en cuenta que la viabilidad de esta integración está sujeta a las diferencias existentes entre los modelos de datos de los dos sistemas.

También se podrían realizar ajustes para extender la funcionalidad del modelo y el prototipo desarrollado, a fin de soportar la generación de horarios de todas las facultades de la Universidad Distrital Francisco José de Caldas. La conversión de los algoritmos desarrollados en frameworks que permitan la especificación de las restricciones a evaluar, permitiría la generalización del modelo para alcanzar dicho objetivo y ser implementado en distintas universidades. Aunque una conversión como esta requiere de un arduo trabajo, se podrían incluir las restricciones relacionadas con los estudiantes para así medir y observar el comportamiento de los algoritmos, y conseguir una evolución en el trabajo desarrollado.

Finalmente, se podría modificar el cálculo de la restricción jornadas contrarias debido a que es la restricción que más violaciones presenta. Para tal fin se debería agregar la capacidad de determinar en qué jornada se encuentran la mayoría de los grupos y calcular la violaciones contra dicha jornada y no respecto al número del segmento, teniendo en cuenta que siempre se debe dar la posibilidad al estudiante de tomar sus asignaturas en alguna de las dos jornadas.

7 GLOSARIO

ACO: Algoritmo de Colonia de Hormigas (Ant Colony Optimization).

Actualización de Feromona: Metodología propia del algoritmo de Colonia de hormigas utilizada para aumentar la probabilidad en la selección de una solución prometedora.

ANOVA (Analysis of Variance): Análisis de la varianza. Es método estadístico que permite medir si existen diferencias entre dos o más muestras (M. Lane, 2011).

Asignatura: Materia que será dictada por el docente y que hace parte del programa de estudios.

Asignatura Fraccionable: Asignatura que puede ser dictada por 2 docentes.

ATS: Algoritmo de Búsqueda Tabú Adaptativa (Adaptative Tabu Search).

Candidato: Conjunto de movimientos generados a partir de la modificación de las asignaciones realizadas a un grupo.

Carga académica especificada: número de horas que se deben asignar a un docente.

Colisión: Se presenta cuando un docente o salón posee más de una asignación en una misma franja.

Criterio de aspiración: condición que se debe cumplir para que un candidato de la lista tabú pueda ser seleccionado.

Cromosoma: Solución utilizada por GA en la operación de cruce y mutación.

Cruce: Es el intercambio de genes entre dos cromosomas.

Espacio Académico: Agrupador de los recursos académicos grupo, docente, franja y salón. Siempre estará conformado por un grupo, pero podrá carecer del resto de recursos.

Evaporación de Feromona: Metodología propia del algoritmo de Colonia de hormigas utilizada para disminuir la probabilidad de escogencia de una solución poco prometedora.

Feromona: Estructura Utilizada para guiar al algoritmo de Colonia de hormigas hacia una solución óptima.

Franja: Recurso académico que determina la hora inicial, la hora final y el día en que se ha programado un espacio académico. Así mismo, un conjunto de franjas determina la disponibilidad horaria de docentes y salones.

GA: Algoritmo Genético (Genetic Algorithm).

Gen: Espacio Académico utilizado por GA en las operaciones de cruce y mutación.

Grupo: Es la concreción de una asignatura. Una asignatura estará conformada por uno o más grupos a los cuales se inscriben un estudiante. Los grupos de una misma asignatura heredan sus características (intensidad horaria y número de semestre) pero pueden diferir en cuanto a horario y docente asignados, y número de estudiantes inscritos.

Grupo fraccionable: Grupo que pertenece a una asignatura fraccionable.

Lista Tabú: Lista que almacenará los candidatos que no pueden ser seleccionados por el algoritmo TS a menos que superen un criterio de aspiración.

Material genético: Son los objetos que componen un espacio académico (Docente, Grupo, Franja, Salón).

Movimiento: Espacio académico generado a partir de la modificación de uno o varios recursos que conforman otro espacio.

Mutación: El cambio de material genético de un gen, elegido de forma aleatoria

Número de genes a cruzar: Es una lista de genes candidatos a cruzar obtenido de uno de los cromosomas padres.

Población Inicial: Número de individuos (cromosomas) generados desde el inicio por el Algoritmo Genético haciendo uso del Algoritmo Greedy.

Profundidad: Determina la cantidad de movimientos que TS creará a partir de un grupo seleccionado.

Recurso Académico: Cada uno de los elementos que conforman un espacio académico: grupo, docente franja y salón.

Recurso Pedagógico: Recursos que pueden ayudar en el proceso de enseñanza del docente (Televisores, Video Beams, Persianas).

Segmento*:* Conjunto de espacios académicos conformado por el n-ésimo grupo de cada una de las asignaturas que pertenecen a un mismo semestre. Cada semestre estará compuesto por varios segmentos, de la siguiente forma:

Segmento 1 = {grupo 1 Calculo **1**, grupo 1 Física **1**,…., Grupo 1 Catedra **1**}
Segmento 2 = {grupo 2 Calculo **1**, grupo 2 Física **1**,…., Grupo 2 Catedra **1**}
Segmento 3 = {grupo 1 Calculo 2, grupo 1 Física 2,…., Grupo 1 Catedra 2}

Donde el segmento 1 y 2 pertenecen al semestre **1**, mientras el 3 pertenece al segundo semestre.

Set de Pruebas: Conjunto de valores que se establecen para los parámetros propios de cada técnica y con los que se ejecutan las pruebas para medir el rendimiento de los algoritmos.

Solución*:* Conjunto de espacios académicos o asignaciones realizadas y su respectiva función objetivo.

Solución factible: Solución en la que se ha generado un horario que puede ser implementado por la universidad.

TS: Algoritmo de Búsqueda Tabú (Tabu Search).

8 BIBLIOGRAFÍA

9th International Conference on the Pratice and Theory of Automated Timetabling. (2012). Recuperado el 22 de Noviembre de 2012, de PATAT 2012: http://www.patat2012.com/

Abramson, D. (1991). Constructing School Timetables using Simulated Annealing: Sequential and Parallel Algorithms. *Management Science, 37*(1), 98-113.

Arito, F. L., Leguizamón, G., & Errecalde, M. (2010). *Algoritmos de Optimización basados en Colonias de Hormigas aplicados al Problema de Asignación Cuadrática y otros problemas relacionados.* Universidad Nacional de San Luis, San Luis.

Astaiza A., L. G. (Diciembre de 2005). Programación de Exámenes: Un enfoque Práctico. *Revista Ingenieria e Investigacion, 25*(3), 92-100.

Aycan, E., & Ayav, T. (2009). *Solving the Course Scheduling Problem Using Simulated Annealing.* Izmir Institute of Technology, Department of Computer Engineering. Patiala: IEEE.

Ayob, M., & Jaradat, G. (2009). Hybrid Ant Colony Systems For Course Timetabling Problems. *2nd Conference on Data Mining and Optimization* (págs. 120-126). Selangor, Malasia: University Kebangsaan.

Bell, E. J., & McMullen, R. P. (Julio de 2004). Ant colony optimization techniques for the vehicle routing problem. *Advanced Engineering informatics*, 41-48.

Brownlee, J. (2014). *Clever Algorithms.* Recuperado el 04 de 2015, de http://www.cleveralgorithms.com/nature-inspired/stochastic/tabu_search.html

Caballero Mejia, J. M. (2008). *Asignación de Horarios de Clases Universitarias mediante algoritmos evolutivos.* Tesis de Maestría, Universidad del Norte, Ingeniería Industrial, Barranquilla.

Castro Medina, E., & Medaglia, A. (2004). *Heurística basada en programación entera binaria para el problema de asignación de salones en la Universidad de los Andes.* Tesis de Maestría, Universidad de los Andes, Departamento de Ingeniería Industrial, Bogotá, Colombia.

Chacón Montes, P. (1995). Algoritmos Evolutivos: algoritmos genéticos y cuasiespecie. En M. Fernández Graciani, & I. Ramos Salavert, *Vida Artificial* (pág. 90). Castilla, España: Univ de Castilla La Mancha.

Consejo Superior Universitario. (15 de Noviembre de 2002). *Universidad Distrital*. Obtenido de http://acreditacion.udistrital.edu.co/resoluciones/estatuto_docente.pdf

CTIT Institute. (2011). *International TimeTabling Competition*. Recuperado el 22 de Noviembre de 2012, de ICT2011: http://www.utwente.nl/ctit/hstt/itc2011/welcome/

Dengiz, B., & Alaba, C. (2000). *A Tabu Search Algorithm for Computer Networks Design.* Maltepe Ankara, Turquía: Department of Industrial Engineering, Gazi University.

Devore, J. (2008). *Probabilidad Y Estadistica Para Ingenierias Y Ciencias.* Cengage Learning Editores.

Dorigo, M. (1992). Optimization, Learning and Natural Algorithms.

Dorigo, M., & Di Caro, G. (1999). The Ant Colony Optimization Meta-Heuristic. En M. Dorigo, & G. Di Caro, *New Ideas in Optimization* (págs. 11-32). London, UK: McGraw Hill.

Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant Colony Optimization: Artificial Ants as a Computational Intelligence Technique. Bruselas.

Garces Ruiz, A., Granada Echeverri, m., & Gallego, R. (2005). *Balance de fases usando colonia de hormigas.* Pereira.

Glover, F. (1986). Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers And Operations Research, 5*, 533-549.

Glover, F., & Laguna, M. (1997). *Tabu Search.* Norwell, Massachusetts: Springer.

Glover, F., Laguna, M., & Martí, R. (2007). Principles of Tabu Search. En T. F. Gonzales, *Approximation Algorithms and Metaheuristics Chapman & Hall/CRC Computer & Information Science Series.* Taylor & Francis.

Gómez Toro, J. A., Vanegas Castellanos, J. D., & Zuluaga Gómez, N. (2009). *Diseño e Implementación de un Algoritmo para dar Solucion al Problema de Asignacion de Salones (Timetabling) Usando el Metodo de Colonia de Hormigas.* Pereira.

Grefenstette, J., & Baker, J. (1989). How Genetic algorithms Work: A Critical Look at Implicit Parallelism. *Third International Conference on Genetic Algorithms*, 20-27.

Guerra Cubillos, M. A., Pardo Quiroga, E. H., & Salas Ruiz, R. E. (2013). Problema de Scholl Timetabling y algoritmos genéticos: una Revisioón. *Vinculos*.

Györi, S., Petres, Z., & Várkonyi-Kóczy, A. (Julio de 2001). Genetic Algorithms in Timetabling. A New Approach. Budapest, Hungría: Hungarian Society of IFSA.

Hernandez, R., Miranda, J., & Rey, P. (2008). Programación de Horarios de Clases y Asignacion de Salas para la Facultad de Ingeniería de la Universidad Diego Portales Mediante un Enfoque de Programación Entera. *Revista Ingenieria de Sistemas, XXII*, 121-141.

Holland, J. H. (1992). *Adaptation an Natural Artificial System.* Cambrige: MIT Press.

Jaramillo Villegas, E. J. (2014). *Analisis y Diseño de Algoritmos*. Recuperado el 24 de 05 de 2014, de http://www.virtual.unal.edu.co/cursos/sedes/manizales/4060024/Lecciones/Capitulo%20III/ppropuestos.htm

Jaramillo Villegas, E. J. (s.f.). *Analisis y Diseño de Algoritmos*. Recuperado el 24 de 05 de 2014, de

http://www.virtual.unal.edu.co/cursos/sedes/manizales/4060024/Lecciones/Capitulo%20III/ppropuestos.htm

Kim, J. (1997). *Iterated Grid Search Algorithm on Unimodal Criteria.* Blacksburg, Virginia: Virginia Polytechnic Institute and State University.

Kolahan, F., & Doughabadi, M. (2012). The effects of parameter settings on the performance of Genetic. *Advanced Materials Research Vols. 433*, 5994-5999.

Lien-Fu, L., Nien-Li, H., Liang-Tsung, H., & Tien-Chun, C. (2006). An Artificial Intelligence Approach to Course Timetabling. Taiwan.

Lourenco, H. R., Martin, O., & Stützle, T. (2003). *Iterated Local Search, Handbook of Meta-heuristics.* Berlin: Springer-Verlag.

Lü, Z., & Hao, J.-K. (2008). Adaptive Tabu Search for course timetabling. *European Journal of Operational Research*, 235-244.

M. Lane, D. (2011). *Online Statistics Education: An Interactive Multimedia Course of Study.* Recuperado el 1 de Junio de 2015, de http://onlinestatbook.com/2/analysis_of_variance/anova.pdf

Martínez Ruiz, F. J., Sánchez García, E., Muñoz Arteaga, J., & Castañeda Ramírez, C. H. (2006). *Timetabling académico usando algoritmos genéticos y programación celular.* Universidad Autónoma de Zacatecas, Ingeniería en Computación, Zacatecas, Mexico.

Melanie, M. (1998). An Introduction to Genetic Algorithms,. En M. Michell, *An Introduction to Genetic Algorithms* (pág. 9). Massachusetts, Estados Unidos: MIT Press.

Melián Batista, B., & Glover, F. (2003). Introducción a la búsqueda tabú. *Revista Iberoamericana de Inteligencia Artificial, 7*(19), 29-48.

Mendez Giraldo, G. A., Caballero Villalobos, J. P., & Álvarez Pomar, L. (2007). Identificación y programación de asignaturas a ofrecer en un programa de especialización. *Revista Científica y Tecnológica de la Facultad de Ingeniería, 11*(2), 80-87.

MOSAIC Project Services Pty Ltd. (4 - 6 de Abril de 2006). *A Brief History of Scheduling.* Recuperado el 15 de 11 de 2012, de http://www.mosaicprojects.com.au/Resources_Papers_042.html

Muniswamy, V. (2009). *Design and Alalysis of Algorithms.* I. K. International Pvt Ltd.

NetWork, Metaheuristics. (2002). *International Timetabling Competition*. Recuperado el 22 de Noviembre de 2012, de ITC2002: http://www.idsia.ch/Files/ttcomp2002/

Oficina de Planeacion. (2012). Obtenido de http://ingenieria1.udistrital.edu.co/matriz

Patat. (s.f.). *Patat Conferences*. Recuperado el 22 de Noviembre de 2012, de http://www.asap.cs.nott.ac.uk/external/watt/patat/

Peñuela, C. A., Franco B., J. F., & Toro O., E. M. (Junio de 2008). Colonia de hormigas aplicada a la programación óptima de horarios de clase. *Scientia et Technica*(38), 49-54.

Pérez de la Cruz, C., & Ramírez Rodríguez, J. (2011). un algoritmo genético para un problema de horarios con restricciones especiales. *Revista de Matemática: Teoría y Aplicaciones, 18*(2), 215-229.

Pertuz Montenegro, J. A., & Rojas Silva, K. (2007). *Formulación y evaluación de un algoritmo, basado en la meta-heurística "Búsqueda Tabú" para la optimización del ruteo de vehículos con capacidad.* Bucaramanga: Universidad Distrital de Santander.

Phuc, N., Khang, N. T., & Nuong, T. T. (2011). A New Hybrid GA-Bees Algorithm for a Real-world University Timetabling Problem. *Institute of Electrical and Electronics Engineers*, 321 - 326.

Pitol Reyes, F. J. (2011). *Uso de algoritmos evolutivos para resolver el problema de asignación de horarios escolares en la facultad de Psicología de la Universidad Veracruzana.* Tesis de Maestria en Inteligencia Artificial, Universidad Veracruzana, Veracruz, México.

Queen´s University Belfast. (2007). *International Timetabling Competition*. Recuperado el 22 de Noviembre de 2012, de ITC2007: http://www.cs.qub.ac.uk/itc2007/

Riojas Cañari, A. C. (2005). *Conceptos, algoritmo y aplicación al problema de las N-Reinas.* Lima.

Rios, D. (2013). *NeuroIA Artificial Neuronal Network*. Recuperado el 8 de Febrero de 2013, de NeuroIA Artificial Neuronal Network: http://www.learnartificialneuralnetworks.com/geneticalg.html#intro

Rodriguez Garcia, j. (2010). *Analisis de algoritmos basados en colonia de hormigas en problemas de camino mínimo.*

Saldaña Crovo, A., Oliva San Martín, C., & Pradenas Rojas, L. (2007). Modelos de Programación Entera para un Problema de Programación de Horarios para Universidades. *Revista Chilena de Ingenieria, 15*, 245-259.

Saremi, A., ElMekkawy, T., & Wang, G. (2007). *Tuning the Parameters of a Memetic Algorithm to Solve Vehicle Routing Problem with Backhauls Using Design of Experiments.* Winnipeg, MB, Canada: Department of Mechanical & Manufacturing Engineering, University of Manitoba.

Socha, K. (2003). Max-Min Ant System for international timetabling Competition. *International Timetabling Competition.* Bruselas, Bélgica.

Soria Alcaraz, J. A., Carpio, M., & Puga, H. (2010). A new approach of Design for the Academic Timetabling problem through Genetic Algorithms. *Electronics, Robotics and Automotive Mechanics Conference* (págs. 96-101). Leon Guanajuato: Leon Institute of Technology.

Stutzle, T., & Holger H., H. (2000). *MAX-MIN Ant System.* Bruselas.

Wang, X. (2004). Applying genetic algorithms with extended fitness to deceptive trap functions. *Department of Computer Science*, 1-97.

Wren, A. (1996). Scheduling, Timetabling and Rostering – A Special Relationship? In E. Burke, & P. Ross, *The Practice and Theory of Automated Timetabling: Selected Papers* (pp. 46-75). Springer.

Zhang, L., & Lau, S. (2005). Constructing university timetable using constraint satisfaction programming approach. *International Conference on Computational Intelligence for Modelling, Control and Automation* (págs. 55-60). Wollongong, Australia: Universidad de Wollongong.

9 ANEXOS

9.1 ARQUITECTURA DEL APLICATIVO

9.1.1 DIAGRAMAS DE CASOS DE USO

Se presentan a continuación las ilustraciones de los casos de uso pertenecientes a los requerimientos funcionales

9.1.1.1 *Diagrama De Caso De Uso para Gestión de la configuración inicial.*

El conjunto de casos de uso enunciados a continuación, gestiona la configuración inicial para poder ingresar los datos en "Gestión de información", también se encuentra la gestión de la pre-asignación que hace la asignación de un grupo, a un docente en horas específicas. Ejemplo: Orlando Villanueva dicta historia a través del cine, sábados de 8am a 10am.

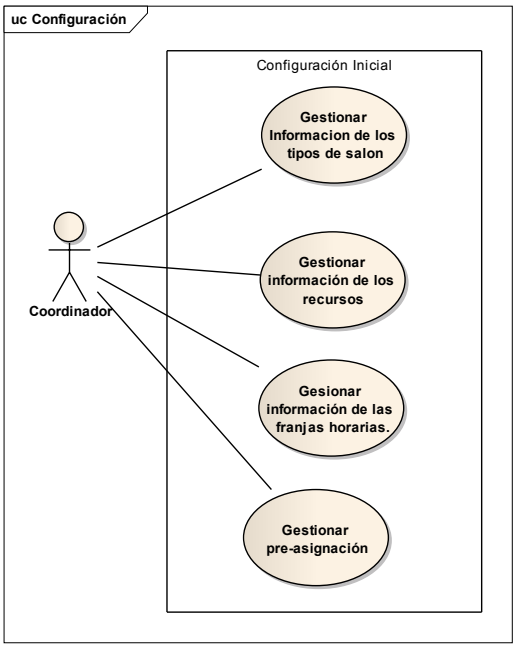


Ilustración 51: *Casos de uso para la gestión de la configuración inicial.*

Nombre Caso de Uso	Gestionar Información de los tipo de salón
Actores	Coordinador
Tipo	Primario
Descripción	El usuario, crea, consulta, actualiza y elimina los datos para los tipos de salón que existen el proyecto curricular de Ingeniería de Sistemas.
Pre-Condición	El aplicativo debe estar en ejecución.
Post-Condición	La información quedará en la base de datos, como suministro inicial para la creación de Salones.

Nombre Caso de Uso	Gestionar Información de los recursos
Actores	Coordinador
Tipo	Primario
Descripción	El usuario, crea, consulta, actualiza y elimina los datos de los recursos, que poseen los salones que les fueron asignados al proyecto curricular de ingeniería de sistemas.
Pre-Condición	El aplicativo debe estar en ejecución.
Post-Condición	La información quedará en la base de datos para su posterior uso.

Nombre Caso de Uso	Gestionar Información de las franjas horarias.
Actores	Coordinador
Tipo	Primario
Descripción	El usuario, crea, consulta, actualiza y elimina los datos de las franjas horarias que maneja el proyecto curricular de ingeniería de sistemas.
Pre-Condición	El aplicativo debe estar en ejecución.
Post-Condición	La información quedará en la base de datos, para su uso en el ingreso de la disponibilidad horaria de los docentes y los horarios de disponibilidad de los salones.

Nombre Caso de Uso	Gestionar pre-asignación
Actores	Coordinador
Tipo	Primario
Descripción	El usuario hace una asignación de un Grupo en una franja horaria determinada a un Docente, posibilitando la edición o eliminación.
Pre-Condición	En la Base de datos debe existir información de un Docente, una Asignatura y una Franja.
Post-Condición	Queda asignado un grupo a un docente en la base de datos.

9.1.1.2 *Diagrama De Caso De Uso para la Gestión de la Información.*

Este conjunto de casos de uso, maneja la creación, la edición, la lectura y el borrado de un docente, una asignatura, un salón, una sede, una facultad y un proyecto curricular.

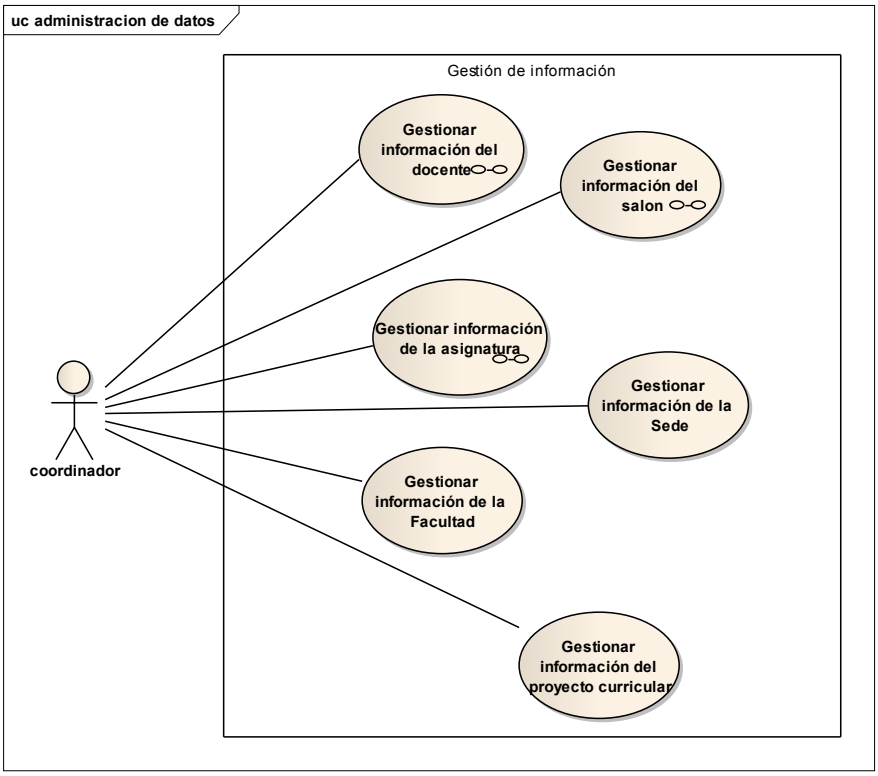


Ilustración 52: *Casos de uso para la gestión de la información de docente, salón, asignatura, sede, proyecto curricular y facultad.*

Nombre Caso de Uso	Gestionar Información del Docente
Actores	Coordinador
Tipo	Primario
Descripción	El usuario, ingresa, consulta, actualiza y elimina los datos de los docentes que están vinculados con el proyecto curricular de ingeniería de sistemas.
Pre-Condición	En la base de datos deben existir datos de franjas horarias.
Post-Condición	En la base de datos queda registrada la información de un docente.

Nombre Caso de Uso	Gestionar Información del salón.
Actores	Coordinador
Tipo	Primario
Descripción	El usuario, crea, consulta, actualiza y elimina los datos de los salones asignados al proyecto curricular de ingeniería de sistemas.
Pre-Condición	En la base de datos deben existir datos de franjas horarias, debe haber una sede registrada.
Post-Condición	En la base de datos queda registrada la información de un salón.

Nombre Caso de Uso	Gestionar Información de la asignatura.
Actores	Coordinador
Tipo	Primario
Descripción	El usuario, crea, consulta, actualiza y elimina los datos de las asignaturas del proyecto curricular de ingeniería de sistemas.
Pre-Condición	En la base de datos debe existir información de tipos de salón.
Post-Condición	En la base de datos queda registrada la información de una asignatura.

Nombre Caso de Uso	Gestionar Información de la sede.
Actores	Coordinador
Tipo	Primario
Descripción	El usuario, crea, consulta, actualiza y elimina los datos de una sede.
Pre-Condición	En la base de datos debe existir información de una facultad.
Post-Condición	En la base de datos queda registrada la información de una sede.

Nombre Caso de Uso	Gestionar Información de la facultad.
Actores	Coordinador
Tipo	Primario
Descripción	El usuario, crea, consulta, actualiza y elimina los datos de una facultad.
Pre-Condición	N-A
Post-Condición	En la base de datos queda registrada la información de una facultad.

Nombre Caso de Uso	Gestionar Información del proyecto curricular.
Actores	Coordinador
Tipo	Primario
Descripción	El usuario, crea, consulta, actualiza y elimina los datos de un proyecto curricular.
Pre-Condición	En la base de datos debe existir información de una sede.
Post-Condición	En la base de datos queda registrada la información de un proyecto curricular.

9.1.1.3 *Diagrama De Caso De Uso para Gestión del Horario.*

El conjunto de casos de uso enunciados a continuación, gestiona la generación, la edición, el borrado y la consulta de un horario, también consultas específicas sobre la carga acdémica de un docente, la distribucion de un salón y la asignación de docentes a una asignatura.

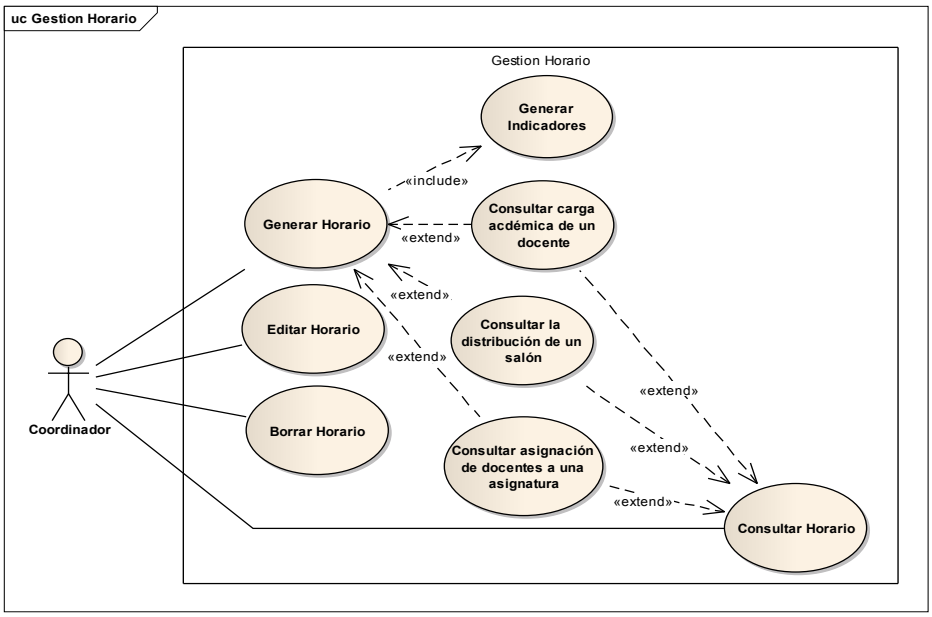


Ilustración 53: *Casos de usos para la gestión del horario.*

Nombre Caso de Uso	Generar Horario
Actores	Coordinador
Tipo	Primario
Descripción	Genera el horario a partir de los datos cargados desde la base de datos, teniendo en cuenta: La disponibilidad horaria del docente, la intensidad horaria de las asignaturas, las características de los salones, como las horas en que los salones están disponibles para el proyecto curricular. Se podrá guardar en la base de datos luego de generado el horario
Pre-Condición	La aplicación debe estar ejecutando y los datos básicos de docentes, salones, asignaturas y grupos deben estar creados y cargados.
Post-Condición	El usuario podrá visualizar el horario o hacer consultas sobre este, como también volver a generar el horario.

Nombre Caso de Uso	Editar Horario
Actores	Coordinador
Tipo	Primario
Descripción	A partir del horario obtenido de la base de datos se podrá hacer modificaciones.
Pre-Condición	El horario generado y almacenado en la base de datos.
Post-Condición	El almacenamiento de los cambios y ajustes sobre el horario en la base de datos.

Nombre Caso de Uso	Borrar Horario
Actores	Coordinador
Tipo	Primario
Descripción	Se eliminará el horario que se almacena en la base de datos.
Pre-Condición	El horario generado y almacenado en la base de datos.
Post-Condición	N-A

Nombre Caso de Uso	Generar Indicadores
Actores	Coordinador
Tipo	Primario
Descripción	Muestra las restricciones violadas y docentes que no fueron asignados.
Pre-Condición	El horario debe estar generado.
Post-Condición	N-A

Nombre Caso de Uso	Consultar carga académica de un docente.
Actores	Coordinador
Tipo	Primario
Descripción	Hará una búsqueda en el horario visualizado y mostrará las horas, salones y asignaturas que impartirá un docente.
Pre-Condición	El horario debe estar generado o en la base de datos.
Post-Condición	El horario del docente se podrá imprimir.

Nombre Caso de Uso	Consultar la Distribución de un salón.
Actores	Coordinador
Tipo	Primario
Descripción	Hará una búsqueda en el horario visualizado, mostrando así las asignaturas, los docentes de esas asignaturas y en qué horas se ocupa el salón en una semana.
Pre-Condición	El horario debe estar creado y visualizado.
Post-Condición	El horario del salón se podrá imprimir.

Nombre Caso de Uso	Consultar asignación de docentes a una asignatura.
Actores	Coordinador
Tipo	Primario
Descripción	Hará una búsqueda en el horario visualizado y mostrará las horas, salones y el docente que dictará un grupo de una asignatura.
Pre-Condición	El horario debe estar generado el horario.
Post-Condición	El horario de la asignatura se podrá imprimir.

Nombre Caso de Uso	Consultar Horario
Actores	Coordinador
Tipo	Primario
Descripción	Se hace una búsqueda de un horario para su posterior visualización, en la base de datos o archivo.
Pre-Condición	Haber Iniciado la aplicación y haber creado un horario.
Post-Condición	Se visualizará el horario y se podrán hacer consultas.

9.1.2 DIAGRAMA DE CLASES DEL MODELO DE NEGOCIO

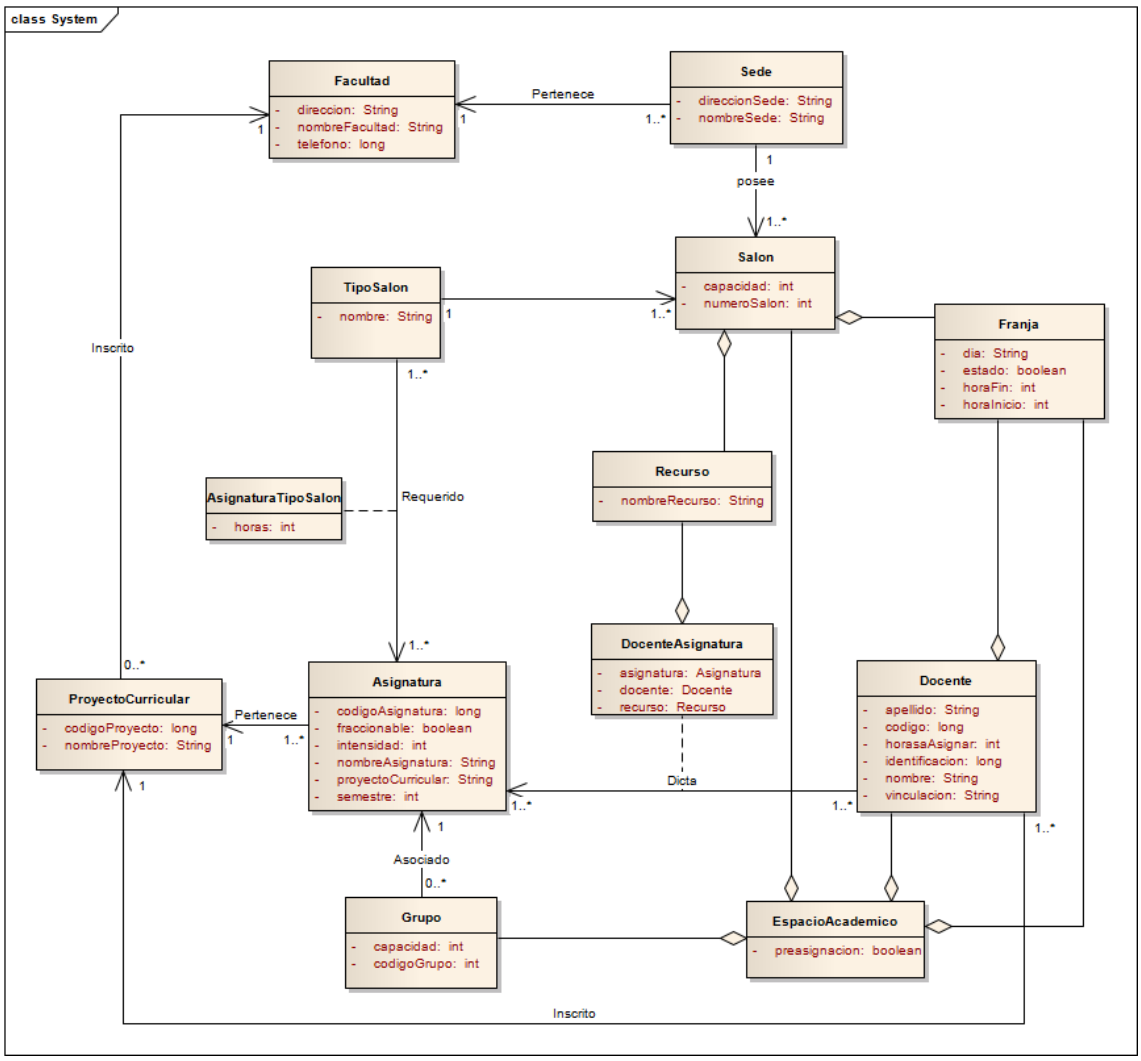


***Ilustración 54**: Diagrama de clases.*

9.1.3 MODELO DE BASE DE DATOS

9.1.3.1 *Generalidades*

Una de las principales características de este proyecto, es el uso de un motor de base de datos para los aspectos referentes a almacenamiento, integridad y persistencia de la información que va ser utilizada por el aplicativo. Para tal fin, se escogió una base de datos Oracle debido a la característica multiplataforma que brinda este motor al igual que el lenguaje de programación Java, el rendimiento obtenido junto con el aplicativo y además para coincidir con la tecnología manejada por el sistema de inscripción de asignaturas de la Universidad Distrital Francisco José de Caldas (Cóndor) y así facilitar posteriores proyectos en los cuales se puedan unificar las soluciones brindadas por estas herramientas.

En cuanto a la tecnología de mapeo que sirve como puente comunicador entre la base de datos relacional utilizada y el aplicativo, se optó por la utilización de JPA (Java Persistence Api), la cual es una herramienta diseñada para aplicaciones Java (SE y EE) en donde se facilita el manejo de operaciones correspondientes a la base de datos como la inserción, eliminación, actualización y consulta de los datos (CRUD).

La base de datos se encuentra diseñada de tal manera que solo existen dos usuarios: el primer usuario es el dueño de todos los objetos almacenados concernientes al aplicativo y además de esto tiene el rol de administrador dentro del sistema. El segundo usuario tiene permisos de escritura, lectura, creación y actualización sobre la información almacenada en las tablas, este es el usuario de base de datos con el cual el aplicativo realiza todas las operaciones.

9.1.3.2 *Modelo Relacional de la Base de Datos*

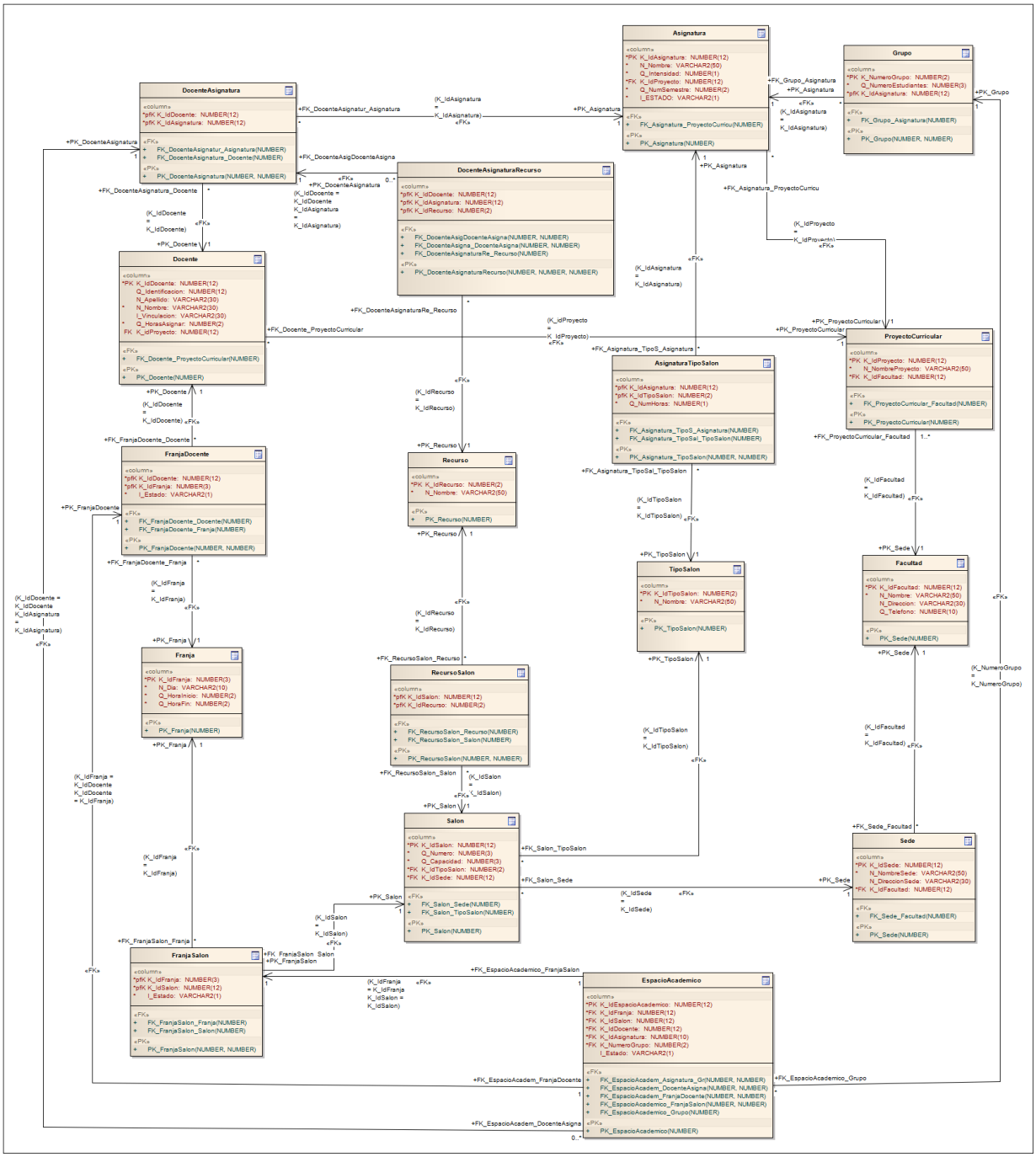


***Ilustración 55**: Modelo Relacional de la Base de datos Oracle del aplicativo.*

9.1.3.3 ***Diccionario de Datos***

Nombre de Tabla / Descripción	Nombre de Atributo	Contenido	Tipo	Formato (Rango)	Tipo de Clave
Asignatura: Representa los datos de una Asignatura en la Universidad	**K_IdAsignatura**	Código que identifica la Asignatura en la base de datos y que corresponde al código de la asignatura en el proyecto curricular de la Universidad.	Number	Numérico de 12 Posiciones sin dígitos decimales	Primaria
	K_IdProyecto	Código que identifica a un proyecto curricular, es el código del proyecto curricular al que pertenece la asignatura.	Number	Numérico de 12 Posiciones sin dígitos decimales	Foránea que referencia a la clave primaria de la tabla ProyectoCurricular
	N_Nombre	Nombre de la Asignatura dentro del proyecto curricular.	Varchar2	Alfanumérico de 50 posiciones	
	Q_Intensidad	Intensidad horaria de una asignatura en el proyecto curricular.	Number	Numérico de 1 posición sin dígitos decimales (1 – 9)	

	Q_NumSemestre	Semestre al cual pertenece la asignatura	Number	Numérico de 2 posición sin dígitos decimales (1 – 12)	
	I_Estado	Bandera que representa si una asignatura puede ser dictada por 2 profesores	Varchar2	Alfanumérico de 1 posición (1, 0)	
AsignaturaTipoSalon Tabla de Rompimiento que almacena los tipos de salón que requiere una asignatura.	**K_IdAsignatura**	Código que identifica a la asignatura en el Proyecto Curricular	Number	Numérico de 12 Posiciones sin dígitos decimales	Foránea que referencia a la clave primaria de la tabla Asignatura (Primaria)
	K_IdTipoSalon	Código que identifica al tipo de salón, hace referencia al tipo de salón que necesita una asignatura.	Number	Numérico de 12 Posiciones sin dígitos decimales	Foránea que referencia a la clave primaria de la tabla TipoSalon (Primaria)
	Q_NumHoras	Número de horas que una asignatura requiere de un tipo de salón	Number	Numérico de 1 posición sin dígitos decimales	

Docente Representa los datos de un docente de la universidad.	**K_IdDocente**	Código que identifica a un docente en el proyecto curricular de la Universidad.	Number	Numérico de 12 Posiciones sin dígitos decimales	Primaria
	K_IdProyecto	Código que identifica a un proyecto curricular, es el código del proyecto curricular al que pertenece el docente.	Number	Numérico de 12 Posiciones sin dígitos decimales	Foránea que referencia a la clave primaria de la tabla ProyectoCurricular
	Q_Identificacion	Identificación de un docente (Cedula de Ciudadanía o cualquier documento que lo identifique).	Number	Numérico de 12 Posiciones sin dígitos decimales	
	N_Apellido	Apellidos del docente.	Varchar2	Alfanumérico de 30 posiciones	
	N_Nombre	Nombres del docente.	Varchar2	Alfanumérico de 30 posiciones	
	I_Vinculacion	Nombre de la vinculación del docente con el proyecto curricular (cátedra, planta, horas, etc.).	Varchar2	Alfanumérico de 30 posiciones	

	Q_HorasAsignar	Número de horas que se necesitan asignar al docente.	Number	Numérico de 2 Posiciones sin dígitos decimales (1 - 60)	
DocenteAsignatura Tabla de rompimiento que almacena las asignaturas que puede dictar un docente.	**K_IdDocente**	Código que identifica a un docente en el proyecto curricular de la Universidad, hace referencia al docente que puede dictar determinada asignatura.	Number	Numérico de 12 Posiciones sin dígitos decimales	Foránea que referencia a la clave primaria de la tabla Docente (Primaria)
	K_IdAsignatura	Código que identifica la Asignatura en la base de datos y asignatura que puede ser dictada por un docente.	Number	Numérico de 12 Posiciones sin dígitos decimales	Foránea que referencia a la clave primaria de la tabla Asignatura (Primaria)
DocenteAsignaturaRecurso Tabla de	**K_IdDocente**	Código que identifica a un docente en el proyecto curricular de la Universidad, hace referencia al recurso requerido por un docente para dictar una asignatura.	Number	Numérico de 12 Posiciones sin dígitos decimales	Foránea que referencia a la clave primaria de la tabla Docente (Primaria)

rompimiento que almacena los recursos necesarios para que un docente dicte una asignatura.	**K_IdAsignatura**	Código que identifica la Asignatura en la base de datos.	Number	Numérico de 12 Posiciones sin dígitos decimales	Foránea que referencia a la clave primaria de la tabla Asignatura (Primaria)
	K_IdRecurso	Recurso que es requerido por un docente para dictar una asignatura.	Number	Numérico de 2 Posiciones sin dígitos decimales	Foránea que referencia a la clave primaria de la tabla Recurso (Primaria)
EspacioAcademico Representa los datos de los espacios académicos de la universidad en un semestre determinado.	**K_IdEspacioAcademico**	Código que identifica a un espacio académico (Secuencia de base de datos).	Number	Numérico, hasta el máximo valor de una secuencia Oracle	Primaria
	K_IdFranja	Código que identifica a una franja en el aplicativo (Secuencia de base de datos).	Number	Numérico cíclico (1 - 999)	Foránea que referencia a la clave primaria de la tabla Franja

	K_IdSalon	Código que identifica a un salón en el aplicativo (Secuencia de base de datos).	Number	numérico, hasta el máximo valor de una secuencia Oracle	Foránea que referencia a la clave primaria de la tabla Salón
	K_IdDocente	Código que identifica a un docente en el proyecto curricular de la Universidad, hace referencia al docente que dicta un espacio académico.	Number	Numérico de 12 Posiciones sin dígitos decimales	Foránea que referencia a la clave primaria de la tabla Docente
	K_IdAsignatura	Código que identifica la Asignatura en la base de datos y que corresponde al código de la asignatura en el proyecto curricular de la Universidad.	Number	Numérico de 12 Posiciones sin dígitos decimales	Foránea que referencia a la clave primaria de la tabla Docente
	K_NumeroGrupo	Código con el cual se identifica un grupo de una asignatura en el proyecto curricular de la Universidad.	Number	Numérico de 2 Posiciones sin dígitos decimales	Foránea que referencia a la clave primaria de la tabla Grupo
	I_Estado	Bandera que hacer referencia a si un espacio académico es producto de una pre asignación o no.	Varchar2	Alfanumérico de 1 posición (1, 0)	

Facultad Representa los datos de una facultad de la Universidad.	**K_IdFacultad**	Código que identifica a la facultad en la Universidad	Number	Numérico de 12 Posiciones sin dígitos decimales	Primaria
	N_Nombre	Nombre de la facultad de la universidad	Varchar2	Alfanumérico de 50 posiciones	
	N_Direccion	Dirección de la facultad en la ciudad o departamento.	Varchar2	Alfanumérico de 30 posiciones	
	Q_Telefono	Número telefónico de la facultad.	Number	Numérico de 10 Posiciones sin dígitos decimales	
Franja Representa los datos de una franja horaria.	**K_IdFranja**	Código que identifica a una franja en el aplicativo (Secuencia de base de datos).	Number	Numérico cíclico (1 - 999)	Primaria
	N_Dia	Día de la semana de la franja.	Varchar2	Alfanumérico de 10 posiciones	
	Q_HoraInicio	Hora de inicio de la franja en el día señalado.	Number	Numérico de 2 Posiciones sin dígitos decimales	

	Q_HoraFin	Hora de finalización de la franja en el día señalado.	Number	Numérico de 2 Posiciones sin dígitos decimales	
FranjaDocente Tabla de rompimiento que almacena las franjas disponibles de un docente.	**K_IdDocente**	Código que identifica a un docente en el proyecto curricular de la Universidad, hace referencia a las franjas disponibles por un docente.	Number	Numérico de 12 Posiciones sin dígitos decimales	Foránea que referencia a la clave primaria de la tabla Docente
	K_IdFranja	Código que identifica a una franja en el aplicativo (Secuencia de base de datos), franja en la que está disponible un docente.	Number	numérico cíclico (1 - 999)	Foránea que referencia a la clave primaria de la tabla Franja
	I_Estado	Bandera que hacer referencia al estado de una franja de un docente (disponibilidad de la franja).	Varchar2	Alfanumérico de 1 posición (1, 0)	
FranjaSalon Tabla de rompimiento que	**K_IdFranja**	Código que identifica a una franja en el aplicativo (Secuencia de base de datos), franja en la que está disponible un salón.	Number	numérico cíclico (1 - 999)	Foránea que referencia a la clave primaria de la tabla Franja (Primaria)

almacena las franjas en las que está disponible un salón.	**K_IdSalon**	Código que identifica a un salón en el aplicativo (Secuencia de base de datos).	Number	numérico, hasta el máximo valor de una secuencia Oracle	Foránea que referencia a la clave primaria de la tabla Salón (Primaria)
	I_Estado	Bandera que hacer referencia la disponibilidad de una franja de un salón.	Varchar2	Alfanumérico de 1 posición (1, 0)	
Grupo Representa los datos de un grupo de una asignatura que pertenece al proyecto curricular.	**K_NumeroGrupo**	Código con el cual se identifica un grupo de una asignatura en el proyecto curricular de la Universidad.	Number	Numérico de 2 Posiciones sin dígitos decimales	Primaria
	K_IdAsignatura	Código que identifica la Asignatura en la base de datos y a la asignatura a la que pertenece el grupo.	Number	Numérico de 12 Posiciones sin dígitos decimales	Foránea que referencia a la clave primaria de la tabla Grupo (Primaria)
	Q_NumeroEstudiantes	Número máximo de estudiantes que son programados para asistir a un grupo de una asignatura.	Number	Numérico de 3 Posiciones sin dígitos decimales	
ProyectoCurricular Representa los datos	**K_IdProyecto**	Código que identifica a un proyecto curricular en la Universidad.	Number	Numérico de 12 Posiciones sin dígitos decimales	Primaria

de un proyecto curricular de la Universidad.	**K_IdFacultad**	Código que identifica a la facultad en la Universidad, hace referencia a la facultad a la que pertenece un proyecto curricular.	Number	Numérico de 12 Posiciones sin dígitos decimales	Foránea que referencia a la clave primaria de la tabla facultad
	N_NombreProyecto	Nombre del Proyecto Curricular de la Universidad.	Varchar2	Alfanumérico de 50 posiciones	
Recurso Representa los datos de un recurso asignado a un salón.	**K_IdRecurso**	Código que identifica al recurso dentro del aplicativo (secuencia de base de datos).	Number	Numérico cíclico de 1 a 99	Primaria
	N_Nombre	Nombre del recurso.	Varchar2	Alfanumérico de 50 posiciones	
RecursoSalon Tabla de rompimiento que almacena los recursos que posee un salón.	**K_IdRecurso**	Código que identifica al recurso dentro del aplicativo (secuencia de base de datos).	Number	Numérico cíclico de 1 a 99	Primaria
	K_IdSalon	Código que identifica a un salón en el aplicativo (Secuencia de base de datos).	Number	numérico, hasta el máximo valor de una secuencia Oracle	Foránea que referencia a la clave primaria de la tabla Salón (Primaria)

Salon Representa los datos de un salón de la Universidad.	**K_IdSalon**	Código que identifica a un salón en el aplicativo (Secuencia de base de datos).	Number	numérico, hasta el máximo valor de una secuencia Oracle	Primaria
	K_IdTipoSalon	Código que identifica al tipo de salón, hace referencia a los tipos de salón existentes.	Number	Numérico de 12 Posiciones sin dígitos decimales	Foránea que referencia a la clave primaria de la tabla TipoSalon
	K_IdSede	Código que identifica a una sede en el aplicativo (Secuencia de base de datos).	Number	numérico, hasta el máximo valor de una secuencia Oracle	Foránea que referencia a la clave primaria de la tabla Sede
	Q_Numero	Número que identifica a un salón dentro de una sede de la Universidad.	Number	Numérico de 3 Posiciones sin dígitos decimales	
	Q_Capacidad	Número que especifica la capacidad de un salón de una sede de la Universidad.	Number	Numérico de 3 Posiciones sin dígitos decimales	

Sede Representa los datos de una sede de la Universidad.	**K_IdSede**	Código que identifica a una sede en el aplicativo (Secuencia de base de datos).	Number	Numérico, hasta el máximo valor de una secuencia	Primaria
	K_IdFacultad	Código que identifica a la facultad en la Universidad, hace referencia a la facultad a la que pertenece una sede.	Number	Numérico de 12 Posiciones sin dígitos decimales	Primaria
	N_NombreSede	Nombre de la sede de la Universidad.	Varchar2	Alfanumérico de 50 posiciones	
	N_DireccionSede	Dirección de la sede de la Universidad dentro de la ciudad o un departamento.	Varchar2	Alfanumérico de 30 posiciones	
TipoSalon Representa los datos de un tipo de salón de la Universidad.	**K_IdTipoSalon**	Código que identifica al tipo de salón, hace referencia a los tipos de salón existentes.	Number	Numérico de 12 Posiciones sin dígitos decimales	Primaria
	N_nombre	Nombre del tipo de salón dentro de la Universidad.	Varchar2	Alfanumérico de 50 posiciones	

9.1.3.4 *Reglas de Integridad*

Para el correcto funcionamiento del aplicativo y la estandarización de la información ingresada a la base de datos, se definieron algunas reglas de integridad, las cuales se listan a continuación:

- Las horas a asignar de un docente no deben superar las 60 ni ser iguales o menores a 0.
- El semestre de una asignatura debe ser mayor o igual a 0 (algunas electivas se manejan como semestre 0) y menor o igual a 12.
- Una asignatura no puede tener una intensidad semanal mayor a 8 horas o menor a 0 horas.
- Las franjas de los salones y horarios de disponibilidad de los docentes se rigen por la configuración inicial del aplicativo, y por tal motivo ésta servirá como restricción en el ingreso de los datos. Al definir dicha configuración, el usuario solo puede habilitar los horarios de lunes a sábado entre 6 de la mañana y 10 de la noche.
- Los tipos de salón que podrán ser seleccionados en la creación de salones, deben estar previamente definidos en la configuración inicial del aplicativo.
- Los recursos asociados a los salones y a los requisitos de un docente deben ser definidos en la configuración del aplicativo.

9.1.4 BUENAS PRACTICAS DE PROGRAMACIÓN

El aplicativo desarrollado para dar solución a la problemática planteada en este documento, fue diseñado y elaborado utilizando algunas de las mejores prácticas de programación.

A fin de garantizar la calidad del desarrollo se recurrió a la implementación de las siguientes prácticas y herramientas:

- Manejo de versiones a través del plugin egit que maneja el IDE Eclipse. Este versionamiento fue de extrema importancia para el desarrollo del aplicativo debido a la facilidad que presenta para realizar cambios de versiones, modificaciones y control de cambios en un repositorio común en el cual se encontraban alojados los fuentes del proyecto[10].
- Logging Framework (seguimiento de errores). El manejo de la biblioteca que proporciona Apache Software Foundation fue de gran ayuda para el manejo de

[10] http://subversion.apache.org/

los mensajes de salida del aplicativo en tiempo de ejecución, los cuales facilitan la trazabilidad y el seguimiento en ejecución del código implementado[11].

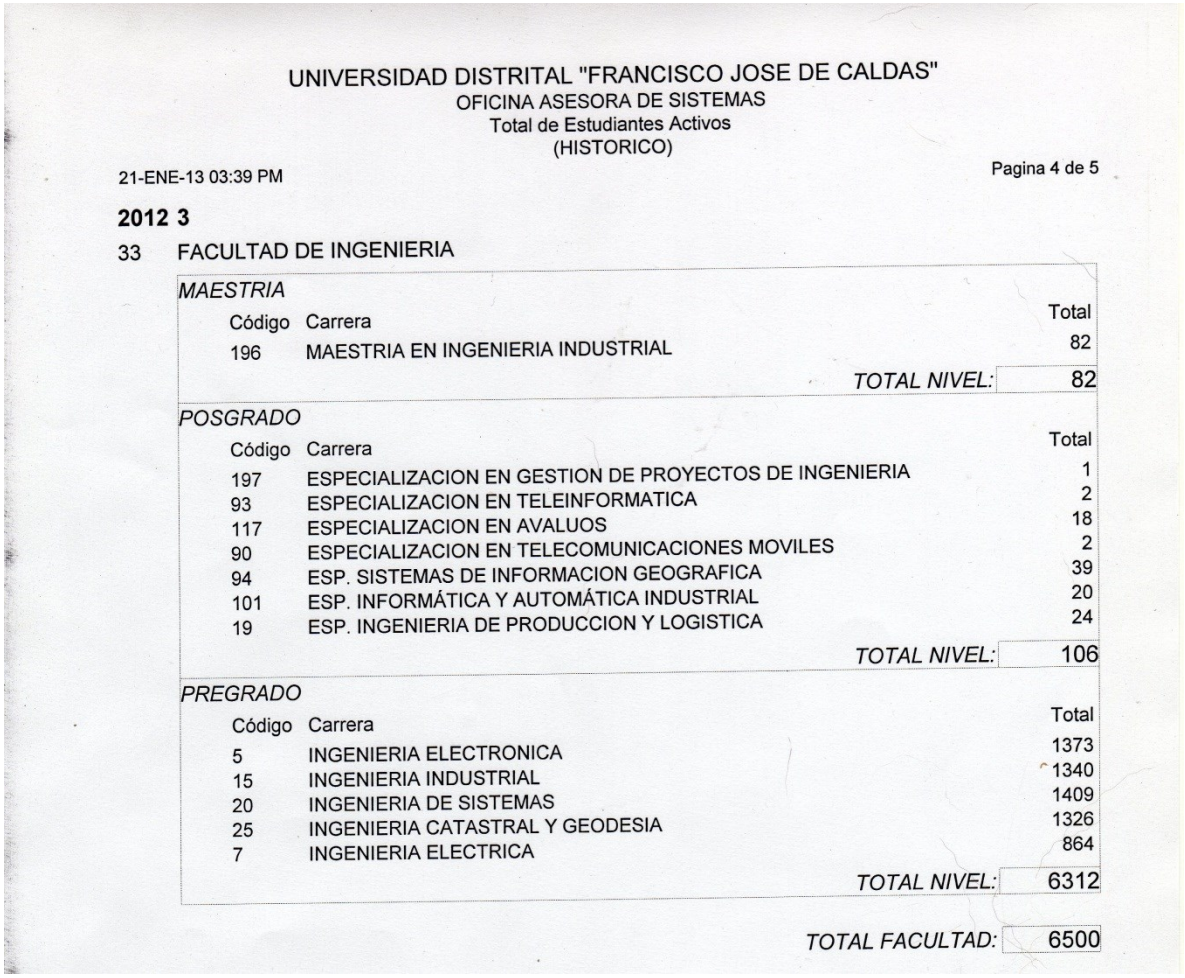
- Fue necesaria la implementación de los patrones de diseño Singleton, Observer, FactoryMethod y Fachada, y el uso del patrón de arquitectura mvc para garantizar la extensibilidad de la aplicación.
- Se realizaron pruebas unitarias a través de JUnit[12] a nivel de la aplicación y del desempeño de cada una de las técnicas desarrolladas.
- Se realizó la documentación del código fuente en un alto nivel de detalle a fin de permitir y facilitar la extensibilidad y la evolución del prototipo en futuros desarrollos.

[11] http://logging.apache.org/log4j/2.x/
[12] http://junit.org/

10 FIGURAS

10.1 ESTUDIANTES ACTIVOS

Total de estudiantes activos de la Facultad de Ingeniería de la Universidad Distrital Francisco José de Caldas periodo académico 2012-III[13].



UNIVERSIDAD DISTRITAL "FRANCISCO JOSE DE CALDAS"
OFICINA ASESORA DE SISTEMAS
Total de Estudiantes Activos
(HISTORICO)

21-ENE-13 03:39 PM — Pagina 4 de 5

2012 3

33 FACULTAD DE INGENIERIA

MAESTRIA

Código	Carrera	Total
196	MAESTRIA EN INGENIERIA INDUSTRIAL	82
	TOTAL NIVEL:	82

POSGRADO

Código	Carrera	Total
197	ESPECIALIZACION EN GESTION DE PROYECTOS DE INGENIERIA	1
93	ESPECIALIZACION EN TELEINFORMATICA	2
117	ESPECIALIZACION EN AVALUOS	18
90	ESPECIALIZACION EN TELECOMUNICACIONES MOVILES	2
94	ESP. SISTEMAS DE INFORMACION GEOGRAFICA	39
101	ESP. INFORMÁTICA Y AUTOMÁTICA INDUSTRIAL	20
19	ESP. INGENIERIA DE PRODUCCION Y LOGISTICA	24
	TOTAL NIVEL:	106

PREGRADO

Código	Carrera	Total
5	INGENIERIA ELECTRONICA	1373
15	INGENIERIA INDUSTRIAL	1340
20	INGENIERIA DE SISTEMAS	1409
25	INGENIERIA CATASTRAL Y GEODESIA	1326
7	INGENIERIA ELECTRICA	864
	TOTAL NIVEL:	6312

TOTAL FACULTAD: 6500

[13] Documento facilitado por la Facultad de Ingeniería de la Universidad Distrital Francisco José de Caldas.