UNIVERSIDAD DE CHILE

FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

"GENERACIÓN DE HORARIOS ACADÉMICOS EN INACAP UTILIZANDO ALGORITMOS GENÉTICOS"

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN TECNOLOGÍAS DE LA INFORMACIÓN

JORGE ANDRÉS AHUMADA AHUMADA

PROFESOR GUÍA:

NELSON BALOIAN TATARYAN

MIEMBROS DE LA COMISIÓN:

BENJAMÍN BUSTOS CARDENAS

PABLO BARCELÓ BAEZA

RODRIGO PAREDES MORALEDA

SANTIAGO DE CHILE

2014

RESUMEN

En todas las instituciones de educación, el proceso de creación de un horario académico es un desafío que se debe sortear semestre a semestre. Este proceso no es simple, ya que está sujeto a restricciones físicas, reglamentarias y legales entre otras.

El objetivo de esta investigación es ofrecer una alternativa de solución para el problema de la asignación de horarios y salas en INACAP, una de las instituciones de educación de mayor envergadura en cuanto a cantidad de alumnos y a infraestructura.

En este trabajo se revisan algunos de los métodos más utilizados para resolver estos problemas de tipo timetabling, un problema considerado de tipo NP-completo, entre los cuales se encuentra la programación lineal y los algoritmos genéticos.

Los algoritmos genéticos, como varios otros métodos, se basan en procesos que se encuentran presentes en la naturaleza. Son técnicas de optimización que emulan de cierta forma los conceptos de la evolución como la supervivencia y la reproducción postulados por Charles Darwin.

En este documento se muestra el diseño y la implementación de un algoritmo genético que logra resolver el problema planteado en tiempos muy razonables y con una muy buena calidad de las soluciones reflejada en el bajo porcentaje de choques horarios.

Se concluye que los algoritmos genéticos son una alternativa muy rápida y confiable para los problemas de optimización

A mis hijas Francisca y Catalina por educarme tanto o más de lo que yo las educo a ellas y por motivar cada pequeño paso que doy en esta vida.

A Luzmira por su amor incondicional y porque cada proyecto que emprendo tiene su empuje y es tan de ella como mío.

A mis madres Gladys y Otilia y mi padre que ya partió, porque nunca olvido donde comenzó esta hermosa historia que es mi vida.

AGRADECIMIENTOS

Agradezco a la Gerencia de Sistemas y Tecnología de INACAP por permitirme desarrollar este proyecto y apoyarme en todos los aspectos necesarios.

Agradezco a la Facultad de Ciencias Físicas y Matemáticas de la Universidad de Chile quien por medio del Departamento de Ciencias de la Computación me entregó una formación que me ha dado valor como profesional, en particular a los profesores, a Christian Bridevaux y Yordy Arévalo por su apoyo y buena disposición.

De forma muy especial agradezco al profesor Nelson Baloian por confiar en mí y ser mi guía en este proyecto.

TABLA DE CONTENIDO

INDICE DE ILUSTRACIONES

1. INTRODUCCIÓN

La creación de horarios académicos en una institución de educación es un procedimiento administrativo que consiste en asignar para cada curso un docente, una sala y una programación semanal. Debido a que en todos los períodos académicos las condiciones varían, esta labor debe desarrollarse al inicio de cada período y muchas veces la información de semestres anteriores no puede utilizarse debido al cambio de condiciones.

Este no es un proceso simple, ya que si se analiza con profundidad, se descubre que existe una gama de factores que se transforman en restricciones que deben ser consideradas al momento de generar un horario académico. Así, encontramos restricciones físicas elementales tales como por ejemplo que una sala no puede ser utilizada por más de un curso a la vez, o que un docente no puede realizar más de una clase en el mismo momento; restricciones reglamentarias, como que un alumno con sus asignaturas al día, en el ámbito universitario, no debe tener coincidencia de horario entre dos de sus cursos; o restricciones legales que impiden que un docente pueda trabajar más de cierta cantidad de horas semanales.

Las características de este proceso lo transforman en un problema de *optimización*, ya que consiste en asignar recursos que siempre son escasos y que deben organizarse de la mejor forma posible. Las instituciones educacionales cuentan con una infraestructura restringida y por ende las salas son limitadas en cuanto a cantidad y capacidad y no siempre han sido diseñadas y construidas para las condiciones actuales. En ocasiones las instituciones de educación crecen en cantidad de alumnos más rápidamente que en infraestructura. Por otra parte, la asignación de docentes también está condicionada por restricciones especiales, muchas veces sujeta a cláusulas contractuales, como por ejemplo los profesores *part-time,* muy frecuentes en la educación chilena, que deben realizar sus clases en varias

instituciones, debiendo considerar su jornada parcial y sus tiempos de traslado. Esto complejiza aún más la creación de horarios académicos.

Los estudiantes frecuentemente se ven obligados a aceptar horarios que no le son cómodos, ya que no optimizan el uso de su tiempo. Es común tener horarios con muchas "ventanas" entre clases, los que significa pérdida de tiempo en desplazamiento, como también una mala asignación de las clases, teniendo muchas veces de forma consecutiva aquellas materias más complejas o se planifican las clases que requieren mayor concentración para una hora en que los alumnos se encuentran más cansados. Además, existen restricciones de tipo reglamentaria que suman dificultad a este proceso, como la existencia de alumnos de jornada diurna o vespertina que solo pueden asistir a clases en un horario que no coincida con sus actividades laborales. Una variable importante a considerar es que el impacto que genera un horario de clases en la formación de los estudiantes amerita que las instituciones destinen recursos y tiempo en buscar la mejor planificación para sus alumnos. Si no se cuenta con la tecnología adecuada, esa búsqueda puede tardar mucho tiempo y la solución puede no ser siempre la óptima.

Esta investigación espera dar solución al problema de la creación de horarios académicos en la institución de educación superior de mayor envergadura en Chile, INACAP, mediante la incorporación de tecnología y el reemplazo del proceso manual ejecutado en la actualidad, en el cual los Directores de Carrera invierten tiempo que podría ser dedicado a otras labores.

El principal objetivo de esta investigación es construir una solución que ayude a reducir el tiempo destinado actualmente en INACAP a la creación de horarios académicos y que además los horarios generados sean óptimos y con menos violación de las restricciones.

Es muy relevante, además, generar una investigación que permita seleccionar la mejor alternativa para resolver el problema de la generación

de horarios de INACAP y el conocimiento adquirido será un activo importante para futuras implementaciones de mejores soluciones a este problema.

Se espera diseñar una solución que entregue un horario válido para cualquier sede de INACAP en un tiempo "razonable" para este tipo de problemas, es decir, que el tiempo que demore haga que su aplicación sea usable en la práctica.

Dado que la institución ya cuenta con una plataforma tecnológica importante, esta solución no puede estar aislada y debe ser parte de ella el desarrollo de un método de traspaso de información que permita integrarla con el sistema SIGA (Sistema Integrado de Gestión Académica).

Es importante que el método desarrollado pueda ser validado, aplicándolo a una situación pasada y comparando el resultado obtenido y el tiempo utilizado con la solución obtenida manualmente.

En el marco de este trabajo se hace mención a los diferentes métodos que han sido utilizados con mayor o menor éxito para resolver el problema, siendo los más recurrentes la programación lineal entera y el uso de algoritmos genéticos.

2. ESTADO DEL ARTE

2.1 Timetabling y Scheduling

En el contexto de la asignación de horarios académicos, existen dos conceptos claves en la literatura que agrupan las investigaciones en torno al problema. Los conceptos con timetabling y scheduling.

Un problema de scheduling es aquel que se trata de la asignación de recursos en el tiempo para llevar a cabo un conjunto de tareas [Baker 1974] y tiene como objetivo minimizar el costo total de los recursos asignados [Wren 1996].

Se define como timetabling el problema de asignar ciertos recursos, sujeto a limitaciones, en un número limitado de horarios y lugares físicos con el objeto de satisfacer una serie de objetivos en el mayor grado posible [Wren 1996].

Los problemas de timetabling contienen restricciones fuertes y restricciones débiles [Larrosa 2003]. Las restricciones fuertes o duras son aquellas que deben ser satisfechas de forma mandatoria y su no cumplimiento inhabilita la solución. Corresponden generalmente a restricciones de tipo espacial o temporal que deben cumplirse. Algunos ejemplos de restricciones fuertes son: no se puede usar simultáneamente un espacio por dos grupos diferentes, no se puede asignar una persona simultáneamente a dos lugares diferentes. Las restricciones débiles o suaves son aquellas que es deseable que se cumplan, pero que su incumplimiento no inhabilita la solución aunque la hace de menor calidad. Generalmente son restricciones de preferencia o de priorización, y muchas veces son estas restricciones las que se desean maximizar (o minimizar según sea el caso) para buscar acercarse a la solución óptima.

Así, el objetivo en un problema de timetabling es minimizar el incumplimiento de las restricciones suaves [Tallabó 1999].

Si bien cada problema de asignación de horarios es diferente a otro, existe una clasificación básica de éstos que facilita su comprensión [Schaerf 1995]:

- Programación de clases y profesores: Considera la asignación de un conjunto de docentes a un conjunto de cursos en un período de tiempo. Este tipo de problema puede considerarse un problema de tipo scheduling.
- Programación de clases y salas: Considera la asignación de un conjunto de cursos a un conjunto de salas en un período de tiempo. Este tipo de problema puede considerarse un problema de tipo timetabling.
- Asignación de horarios de exámenes: Consiste en la calendarización de los exámenes de los alumnos, asignando docentes y salas. Este tipo de problemas son más simples debido a que los exámenes no suponen una persistencia en el tiempo, si no se aplican solo una vez en el período.

Generalmente, los problemas de asignación de horarios académicos corresponden a una suma de los dos primeros tipos: Programación de clases y profesores y Programación de clases y salas. Por ende, la mayoría de los problemas de asignación de horarios tienen componentes de timetabling y de scheduling. Por esto mismo, muchas veces el problema se aborda en dos etapas [Granada 2006], asignando en una primera etapa los docentes a los cursos en un horario y en una segunda etapa la combinación docente-curso a la sala.

2.2 Un problema complejo

La teoría de la computación, mediante la teoría de la complejidad computacional estudia los recursos que necesita un algoritmo para resolver un problema. Los recursos que son objeto de estudio son el tiempo y el espacio. El tiempo se traduce en la cantidad de pasos de ejecución de un

algoritmo para resolver el problema y el espacio se puede definir como la cantidad de memoria necesaria. Desde ese punto de vista los problemas se clasifican en muchas clases, pero en el contexto de este estudio definiremos tres de ellas: P, NP y NP-Completo [Garey 1983].

La clase de complejidad P es el conjunto de los problemas de decisión que pueden ser resueltos en una máquina (de Turing) determinista en tiempo polinómico, lo que corresponde intuitivamente a problemas que pueden ser resueltos aún en el peor de sus casos. La clase de complejidad NP es el conjunto de los problemas de decisión que pueden ser resueltos en tiempo polinómico por una máquina de Turing no-determinista. NP es el acrónimo en inglés de Polinómico No determinista (Non-Deterministic Polynomial-time). La clase NP es importante porque contiene muchos problemas de búsqueda y optimización para los que se desea saber si existe cierta solución o si existe una mejor solución que las conocidas. Todos los problemas de esta clase tienen la propiedad de que, sin embargo, cualquier solución suya puede ser verificada efectivamente. Dada su importancia, se han hecho muchos esfuerzos para encontrar algoritmos que resuelvan algún problema de NP en tiempo polinómico. Aun así, para algunos problemas de NP no es posible encontrar siquiera un algoritmo mejor que simplemente realizar una búsqueda exhaustiva. La clase de complejidad NP-completo es el subconjunto de los problemas de decisión en NP tal que todo problema en NP se puede transformar polinomialmente en cada uno de los problemas de NP-completo. Una transformación polinomial es un algoritmo determinista que transforma instrucciones de un problema en instrucciones del otro. Se puede decir que los problemas de NP-completo son los problemas más difíciles de NP y, muy probablemente, no formen parte de la clase de complejidad P. La razón es que de tenerse una solución polinómica para un problema de NP-completo, todos los problemas de NP tendrían también una solución en tiempo polinómico.

Como consecuencia de esta definición, de tenerse un algoritmo en P para uno de los problemas NP-completos, se tendría una solución en P para todos los problemas de NP [Cook, 1971].

Para el problema de timetabling, al ser un problema de complejidad NP-completo, se hace imposible intentar obtener la mejor solución mediante el descubrimiento de todas las posibles soluciones para los casos en que el tamaño del problema exceda una cierta dimensión, normalmente bastante limitada, por lo cual debe ser abordado utilizando alguno de los siguientes enfoques [Cormen 2001]:

- Aproximación: Un algoritmo que rápidamente encuentra una solución no necesariamente óptima, pero dentro de un cierto rango de error. En algunos casos, encontrar una buena aproximación es suficiente para resolver el problema, pero no todos los problemas NP-completos tienen buenos algoritmos de aproximación.
- Probabilidad: Un algoritmo probabilístico obtiene en promedio una buena solución al problema planteado, para una distribución de los datos de entrada dada.
- Heurísticas: Son algoritmos que no entregan soluciones exactas y que se basan en una estrategia de búsqueda para llegar más rápido a una solución.

El problema de la creación de horarios académicos es un problema clasificado como NP-completo [Gotlieb 1964], lo cual lo convierte en objeto de estudio por su dificultad y sus múltiples variantes.

Este es un problema que en su temática general es universal a todas las instituciones de educación, pero las restricciones particulares hacen que cada instancia requiera una solución propia.

2.3 Alternativas de solución

La complejidad de un problema de timetabling, al querer aplicarlo en un ambiente real con gran cantidad de restricciones y en especial en entornos universitarios donde el tamaño de las variables tales como la cantidad de cursos, cantidad de salas o cantidad de docentes, lo hacen un problema muy estudiado y para el cual existe mucha investigación. A continuación se muestran algunas de las metodologías que se han utilizado para buscar una solución a la asignación de horarios.

2.3.1 Meta-heurísticas

Las metodologías de meta-heurísticas son métodos probabilísticos que parten de una solución inicial y a medida que el algoritmo se va ejecutando, se va mejorando esta solución. Esto métodos se han aplicado con bastante éxito para encontrar soluciones a problemas NP-completos, para los cuales no existe un algoritmo que encuentre su solución óptima en un tiempo razonable. Este tipo de soluciones están basadas en un principio de "aumento sucesivo" e intentan evitar la generación de soluciones de pobre calidad. Las diferencias entre las distintas meta heurísticas existentes tienen que ver con las técnicas que se emplean para evitar que el algoritmo converja en alguna solución sub óptima y el tipo de trayectoria seguida en el espacio de cualquiera de las soluciones parciales o totales. Una de las desventajas que presentan es el alto costo de procesamiento que requieren para llegar a una solución de buena calidad.

Dentro de las meta heurísticas se encuentran los siguientes métodos:

- Búsqueda tabú (Tabu Search).
- Algoritmos genéticos.
- Recocido simulado o enfriamiento lento simulado (simulated annealing).
- Colonia de hormigas (ACO Ant Colony Optimization).

2.3.1.1 Búsqueda Tabú (Tabu Search)

La búsqueda tabú es un método heurístico que proviene de la familia de técnicas denominada "búsqueda local", que está basado en el concepto de "vecindario". Considere un problema de optimización, en que *S* es su espacio de búsqueda (posibles soluciones) y *f* su función objetivo a minimizar. Una función *N*, que depende de la estructura del problema específico, asigna a cada solución posible $s \in S$ su vecindad $N(s) \subseteq S$. Cada solución $s' \in N(s)$ es llamada vecino de *s* [Schaerf 1996].

Las técnicas de búsqueda local comienzan con una solución inicial (que puede ser obtenida por otra técnica o generada aleatoriamente) y a través de ciclos exploran el espacio de búsqueda, pasando iterativamente desde una solución a uno de sus "vecinos", mediante alguna modificación que transforma la solución inicial.

El algoritmo de búsqueda local llamado búsqueda tabú fue diseñado por Glover [Glover 1989], y consiste en un procedimiento de búsqueda local o por vecindades para moverse de forma iterativa desde una solución *s* a una solución *s'* en la vecindad de *s*. Se comienza desde una solución inicial *s0 y* el algoritmo de búsqueda tabú explora un subconjunto de vecinos *N(s)* del espacio de soluciones s; el vecino que presente una mejor función objetivo se convierte en la nueva solución actual, independiente que su función objetivo sea mejor o peor que el valor de la función objetivo en *s*. El método utiliza una lista llamada lista tabú, que es una estructura de memoria de corto plazo que contiene los movimientos que fueron realizados en las últimas *n* iteraciones. Generalmente esta lista es tratada como una cola de largo fijo, es decir cuando un nuevo movimiento es agregado a la lista, el más antiguo es desechado. La búsqueda excluye los movimientos que se encuentran en la lista tabú [Schaerf 1996]. El algoritmo finaliza luego de una cantidad de iteraciones predefinidas o cuando la función objetivo no está sobrepasando un límite de mejora definido como función de aspiración.

Búsqueda tabú constituye un conjunto de técnicas de alto nivel cuyas características como el uso de memoria adaptativa y que puede operar de manera determinista o probabilística, la hacen una muy buena herramienta en el contexto del timetabling. Está diseñado para encontrar soluciones sub óptimas en problemas de optimización combinatoria.

2.3.1.2 Algoritmos genéticos

Los algoritmos genéticos son métodos de búsqueda basados en el mecanismo de selección natural, lo que permite que se explore en busca de posibles soluciones en un espacio de búsqueda mayor al espacio de búsqueda de los métodos tradicionales [Golberg 1989]. Su funcionamiento se construye con dos procesos básicos basados en la teoría de la evolución postulado por Darwin, la selección natural y la reproducción, lo cual hace que las soluciones vayan mejorando a medida que el algoritmo avanza, ya que éstas toman lo mejor de las soluciones anteriores o padres. En el algoritmo genético las soluciones son llamadas cromosomas, para los cuales se debe definir una representación que permite operar con rapidez sobre ellos. Se debe comenzar con un grupo de cromosomas o solución inicial que generalmente es obtenida de forma aleatoria o a través de otro método.

El algoritmo también define una función objetivo que se encarga de entregar una medida de bondad de la solución. La iteración se realiza cruzando soluciones, emulando la reproducción, y mediante la función objetivo seleccionando alguna de las soluciones hijas, no necesariamente las que tengan mejor función objetivo. Para explorar nuevas soluciones los algoritmos genéticos introducen el concepto de mutación que consiste en realizar pequeñas modificaciones, generalmente aleatorias, para explorar distintos espacios de solución y evitar que el algoritmo converja hacia óptimos locales.

Se definen como condiciones de término de estos algoritmos una cantidad de iteraciones o cuando la mejora que se produce en la función objetivo no es sustancial después de un número de iteraciones.

Las aplicaciones más comunes de los algoritmos genéticos han sido los problemas de optimización, para los cuales su comportamiento es muy eficiente y confiable. Sin embargo, no se puede resolver todos los problemas con esta técnica ya que para algunos problemas podría no ser apropiado. Se debe considerar que para que un problema pueda ser resuelto mediante algoritmos genéticos debe tener un espacio de búsqueda limitado dentro de cierto rango, debe poder definirse una función de aptitud que evalúe que tan buena o mala es la respuesta del algoritmo y las soluciones deben codificarse de una forma relativamente fácil.

2.3.1.3 Algoritmos de enfriamiento lento simulado (simulated annealing)

La familia de algoritmos de enfriamiento lento simulado aparece a principios de los años ochenta [Kirkpatrick 1983]. Se trata de un modelo de resolución para la optimización de problemas de tipo combinatorio con mínimos locales y su aproximación consiste en generar de forma aleatoria una solución cercana a la solución actual y aceptarla como buena si consigue reducir la función de costo.

Este método se basa en la emulación del proceso físico de solidificación controlada, por el cual un sólido es calentado hasta que se funda y luego es enfriado lentamente de modo que cuando su estructura se encuentre congelada esto ocurra con el menor consumo de energía y alcance una forma cristalina, sin malformaciones locales, cercana a la perfección [Van Laarhoven 1987].

Este algoritmo se inicia con una solución que se va modificando a medida que el algoritmo avanza o itera. Inicialmente el algoritmo no es tan exigente con las soluciones, es decir acepta soluciones con un valor deficiente según

la función objetivo, pero a medida que itera se va haciendo más exigente con las soluciones.

2.3.1.4 Colonia de hormigas

Es común que para buscar solución a problemas complejos se usen métodos inspirados en la naturaleza. El principio del método de optimización de colonia de hormigas (ACO por sus siglas en inglés, Ant Colony Optimization), se basa en el estudio del comportamiento que utilizan las hormigas para marcar el camino más corto entre su colonia y el alimento. La mayoría de las especies de hormigas son prácticamente ciegas, por lo cual deben utilizar feromonas para marcar las rutas que van utilizando y luego otras hormigas pueden seguir ese rastro para el regreso a la colonia. Se puede decir entonces que las hormigas siguen el camino que tenga mayor probabilidad de uso o aquel cuya marca de feromona es más potente.

En la implementación de un algoritmo de colonia de hormigas, las posibles soluciones son denominadas hormigas artificiales, que puede ser una solución factible o no. Esta alternativa de solución está construida en base a las reglas que emulan el comportamiento de las hormigas reales como son la explotación, exploración y evaporación de feromonas. Al comienzo existe una colonia o un conjunto de soluciones posibles y en cada iteración las hormigas crean alternativas en base a la información recopilada por las soluciones antecesoras. La cantidad de hormigas es un parámetro importante a definir en el diseño del algoritmo.

Se define además una estructura de memoria que almacena datos numéricos que emulan la feromona de las hormigas reales. Esta estructura, llamada rastro de feromona, guarda el grado de aceptación de hormigas o soluciones anteriores para una variable de estado que se encuentra en una alternativa de solución.

Este método se diferencia de otros métodos heurísticos en que la construcción de las soluciones las realiza en base al aprendizaje ganado

anteriormente y que ha sido guardado en la matriz de feromonas, dejando en un segundo plano el impacto que pueda tener esta solución en la función objetivo. Al igual que otros métodos, el algoritmo de colonia de hormigas previene que se pueda converger hacia una solución local, para lo cual existe un proceso denominado evaporación de feromonas, el cual disminuye el rastro de feromona utilizando una función de probabilidad.

Algunas de las ventajas de este método en el problema de asignación de horarios, es que tienen alta probabilidad de encontrar algún óptimo global, pues los métodos convencionales regularmente convergen en óptimos locales. Además, la representación es fácil de analizar y comprender. Entre las desventajas del método de colonia de hormigas está el alto grado de incertidumbre en cuanto al tiempo de convergencia del algoritmo y el alto costo de procesamiento que requiere. Al igual que otros métodos heurísticos no asegura el encontrar soluciones exactas, sino solo soluciones óptimas en un vecindario, que muchas veces son suficientes.

2.3.2 Programación lineal entera

Este método proviene del área de la investigación de operaciones, ciencia que resuelve, entre muchos otros problemas, la asignación de recursos y asignación de tareas buscando el conjunto de asignaciones que optimicen un objetivo planteado, generalmente maximizar utilidad o minimizar costo. La esencia de la investigación de operaciones está en el diseño y la creación de modelos matemáticos.

La programación lineal entera formula los problemas mediante un sistema de inecuaciones lineales y define una función objetivo, que se debe maximizar o minimizar según corresponda. Las variables de la función objetivo se encuentran sujeta una serie de restricciones que tienen distinto peso según su importancia en el problema. Las restricciones son expresadas mediante el sistema de inecuaciones.

Numerosos autores han utilizado modelos de programación entera para resolver los problemas de asignación de horarios en ambientes académicos.

2.3.3 Redes Neuronales

Una red neuronal puede definirse como un sistema de procesamiento de información compuesto por un gran número de elementos de procesamiento (neuronas) profusamente conectados entre sí a través de canales de comunicación [Regueiro 1995]. Estas conexiones establecen una estructura jerárquica y permiten la interacción con los objetos del mundo real tratando de emular al sistema nervioso biológico. A diferencia de la computación tradicional, basada en algoritmos predecibles, la computación neuronal permite desarrollar sistemas que resuelvan problemas complejos cuya formalización matemática es sumamente difícil. Esto se logra gracias a los principios de funcionamiento de las redes neuronales, de los cuales citamos a continuación los cinco más importantes [Hilera 1995]: aprendizaje adaptativo, auto organización, tolerancia a fallos, operación en tiempo real y fácil inserción en la tecnología existente.

En 1995, Mausser et al. describieron una metodología basada en redes neuronales para resolver el problema de programación de horarios de entrevistas y asignación de salas y entrevistadores en la universidad de Waterloo considerando restricciones de disponibilidad de salas y adyacencia de clases (clases consecutivas). Para la modelación se consideró una pre-asignación entre el entrevistador y la sala correspondiente, es decir se consideró un índice k llamado "facility" que representaba un par sala-entrevistador, los cuales estaban asignados antes de resolver el problema. El modelo matemático tenía dos grupos de variables de decisión binarias, la primera llamada Xijk que tomaba el valor 1 si el estudiante i se asociaba al bloque horario j en el par sala-entrevistador k y la otra Yijk tomaba valor 1 si el bloque horario j era el primero utilizado en la entrevista por el estudiante i en el par sala-entrevistador k y/o en bloque horario anterior había una

ventana horaria dentro de la entrevista. Esta metodología asociaba una neurona a cada variable de decisión Xijk y penalizaba el incumplimiento de algunas restricciones suaves. Se concluye que los resultados dependen en gran medida de los pesos utilizados dentro de las distintas penalizaciones por incumplimiento de restricciones suaves y del tamaño del problema [Hernández 2008].

2.3.4 Sistemas expertos

Los sistemas expertos son sistemas que buscan modelar el conocimiento y procedimientos usados por un experto humano para resolver un problema determinado. Es un sistema que simula el proceso de aprendizaje, de memorización, de razonamiento, de comunicación y de acción de un experto humano en una determinada rama o dominio.

Con la ayuda de un sistema experto, personas con poca experiencia pueden resolver problemas que requieren un "conocimiento formal especializado". Los sistemas expertos pueden obtener conclusiones y resolver problemas de forma más rápida que los expertos humanos. Los sistemas expertos razonan pero basándose en un conocimiento adquirido y no tienen sitio para la subjetividad [Pluss 1999].

2.3.5 Case-Based Reasoning

El Case-Based Reasoning es el proceso de solucionar nuevos problemas basándose en las soluciones de problemas anteriores. Estos métodos son relativamente nuevos y utilizan soluciones anteriores para construir una nueva solución a un problema de timetabling. La idea de estos métodos es crear una medida del grado de similitud entre dos soluciones para un problema de timetabling con el fin de encontrar una nueva solución tal que la medida de similitud con respecto a la solución anterior no sobrepase cierto límite, lo que permite encontrar soluciones locales más rápidamente.

2.4 Selección de alternativa de solución

En el ámbito de la asignación de horarios, la programación lineal entera no ha sido efectiva en la resolución de problemas de grandes magnitudes y solo se tienen buenos resultados en entornos pequeños y medianos [Hernández 2008]. Sin embargo, existe un enfoque que indica que al subdividir el problema se puede escalar la solución. Esto muchas veces lleva a obtener buenas soluciones parciales pero no una buena solución completa.

Según las características del problema de INACAP este no sería un método adecuado debido al gran tamaño de los conjuntos a asignar (cursos, salas y docentes). Pero existen algunas sedes que utilizan un enfoque de asignación por carreras en el cual este método podría funcionar adecuadamente. Por esto no se descarta la investigación y evaluación de la programación lineal entera en el futuro.

Los algoritmos genéticos poseen ventajas por sobre otros algoritmos, lo cual hace que sean preferibles para este tipo de problemas.

Algunas de las características de los algoritmos genéticos son:

- Son algoritmos estocásticos. Cada ejecución del algoritmo puede entregar a su vez soluciones distintas.
- Son algoritmos de búsqueda múltiple, por lo cual entregan varias soluciones.
- Son algoritmos que exploran un amplio espacio de soluciones válidas.

Ventajas y desventajas de los algoritmos genéticos [Bull 1993].

- No necesitan conocimientos específicos sobre el problema que intentan resolver.
- Operan de forma simultánea con varias soluciones, en vez de trabajar de forma secuencial como las técnicas tradicionales.

- Cuando se usan para problemas de optimización de una función objetivo, resultan menos afectados por los óptimos locales (falsas soluciones) que las técnicas tradicionales.
- Usan operadores probabilísticos en vez de los típicos operadores determinísticos de las otras técnicas.
- La mayoría de las otras técnicas de optimización son también estocásticas.
- Pueden tardar mucho en converger, o no converger en absoluto, dependiendo en cierta medida de los parámetros que se utilicen tamaño de la población, número de generaciones, etc.
- Pueden converger prematuramente debido a una serie de problemas de diversa índole.

La siguiente tabla muestra una comparación resumida de las características que poseen los algoritmos genéticos con respecto a otros métodos de optimización [Goldberg 1989].

Algoritmos genéticos	**Métodos de optimización tradicionales**
• Trabajan con parámetros codificados, es decir que deben codificarse como cadenas de longitud finita sobre algún alfabeto finito.	• Trabajan directamente con los parámetros.
• Utilizan poblaciones de puntos, es decir, se usa una base de datos de puntos	• Operan sobre puntos individuales, ya que sus movimientos en el espacio de

simultáneamente, de tal forma que la probabilidad de quedar atrapados en óptimos locales se reduce.	búsqueda se hacen de un punto a otro usando reglas de transición determinística. Esto puede ocasionar que se encuentren óptimos locales en lugar de óptimos globales.
• No necesitan conocimientos auxiliares sobre el problema, ya que usan información de la función de evaluación con respecto a los cromosomas.	• Requieren de mucha información auxiliar para trabajar adecuadamente.
• Utilizan reglas de transición probabilísticas.	• Usan reglas determinísticas.

3. MARCO TEÓRICO

3.1 Algoritmos genéticos

Un algoritmo genético es un tipo de algoritmo evolutivo que se conoce a fines de los años 60 tras las investigaciones de John Holland en la Universidad de Michigan [Holland 1975]. Este tipo de algoritmo basa su funcionamiento en mecanismos de selección natural y supervivencia de los individuos más fuertes, emulando los procesos de evolución biológica postulados por Darwin en 1859. La definición formal de un algoritmo genético entregada por Goldberg es la siguiente, "*los algoritmos genéticos son algoritmos de búsqueda basados en la mecánica de selección natural y de la genética natural. Combinan la supervivencia del más apto entre estructuras de secuencias con un intercambio de información estructurado, aunque aleatorizado, para constituir así un algoritmo de búsqueda que tenga algo de las genialidades de las búsquedas humanas*" [Goldberg 1989].

En los algoritmos genéticos los individuos representan una solución al problema y las bases biológicas están presentes al existir operaciones que permiten el intercambio de información entre distintos individuos y la transformación de un individuo para adaptarse de mejor forma al ambiente.

Los mejores individuos, y por ende las mejores soluciones, son seleccionados para reproducirse, generando a su vez nuevas soluciones. Este proceso reproductivo se repite hasta obtener un individuo que se acerque a la solución óptima. En un algoritmo genético bien diseñado, al ir seleccionando los mejores individuos para reproducirse, se asegura que en cada iteración se mejore la solución anterior. Del mismo modo, los individuos que no se adapten correctamente al problema, tendrán menos probabilidades de reproducirse, impidiendo así que en una iteración las posibles soluciones se degraden.

La estrategia de solución mediante algoritmos genéticos es aplicable a problemas de distinta índole, siendo una técnica robusta, que si bien no

siempre encuentra la solución óptima, existe evidencia que encuentra soluciones muy aceptables en tiempos razonables, que lo hacen competir con otros métodos de optimización como la programación lineal entera.

En particular en los problemas de *timetabling* o *scheduling*, los algoritmos genéticos han sido usados con mucho éxito encontrando soluciones muy cercanas a las óptimas y adaptándose muy bien a las condiciones particulares de cada problema.

Existen dos elementos centrales en un algoritmo genético: la representación de los individuos o posibles soluciones, denominados *cromosomas* y la *función objetivo o función de aptitud* que mide si el individuo es apto como solución [Goldberg 1989].

También existen tres operaciones fundamentales dentro de los algoritmos genéticos, las cuales se aplican a los cromosomas, como son las funciones de *selección* que permite elegir los mejores cromosomas para la reproducción; *cruza*, que permite realizar el proceso de reproducción entre dos cromosomas; y la *mutación*, que permite transformar un cromosoma en alguno de sus puntos con tal de explorar espacios de soluciones que serían inalcanzables con la cruza.

3.2 Cromosoma

El cromosoma es la representación de una solución al problema, la cual puede estar compuesta de uno o más genes. El cromosoma se conforma de una o más estructuras de datos que deben contener toda la información relevante del problema. La forma que se defina para el cromosoma es de extrema relevancia para asegurar la rapidez de la ejecución del algoritmo genético y es factor preponderante la velocidad con la cual la función objetivo puede evaluar si el cromosoma es apto o no. Los cromosomas deben ocupar el menor espacio posible y deben ser fáciles de preservar durante la ejecución del algoritmo genético [Beligiannis 2008].

La representación más pura de un cromosoma se realiza de forma binaria (Figura 1), donde se asigna a cada parámetro un número determinado de bits.

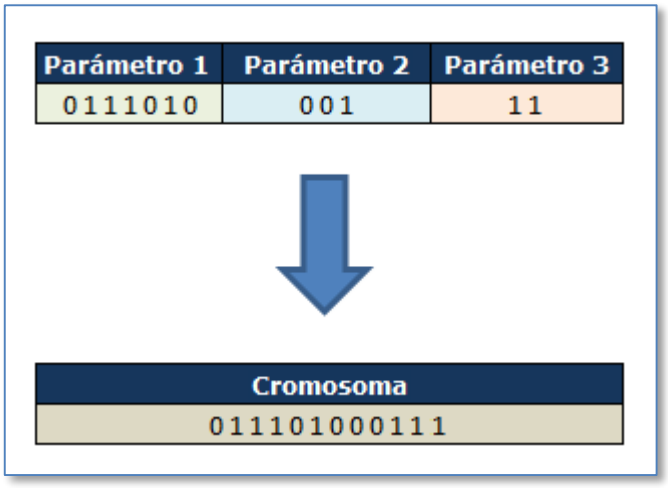


Figura 1: Cromosoma binario

Sin embargo, también pueden existir representaciones con valores no binarios que grafican de mejor forma las soluciones del problema (Figura 2).

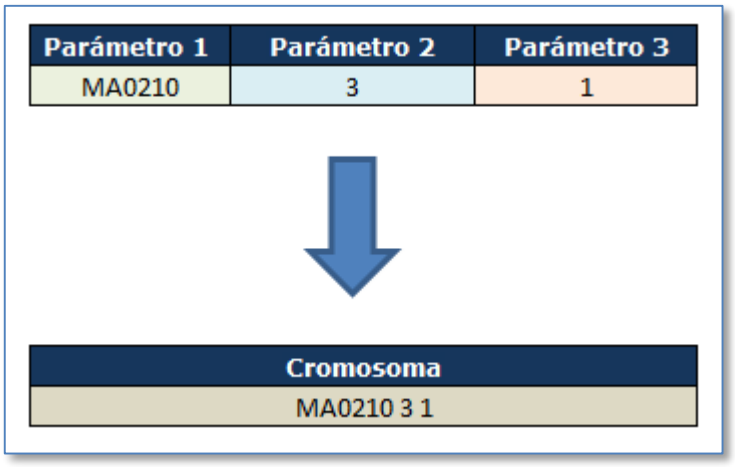
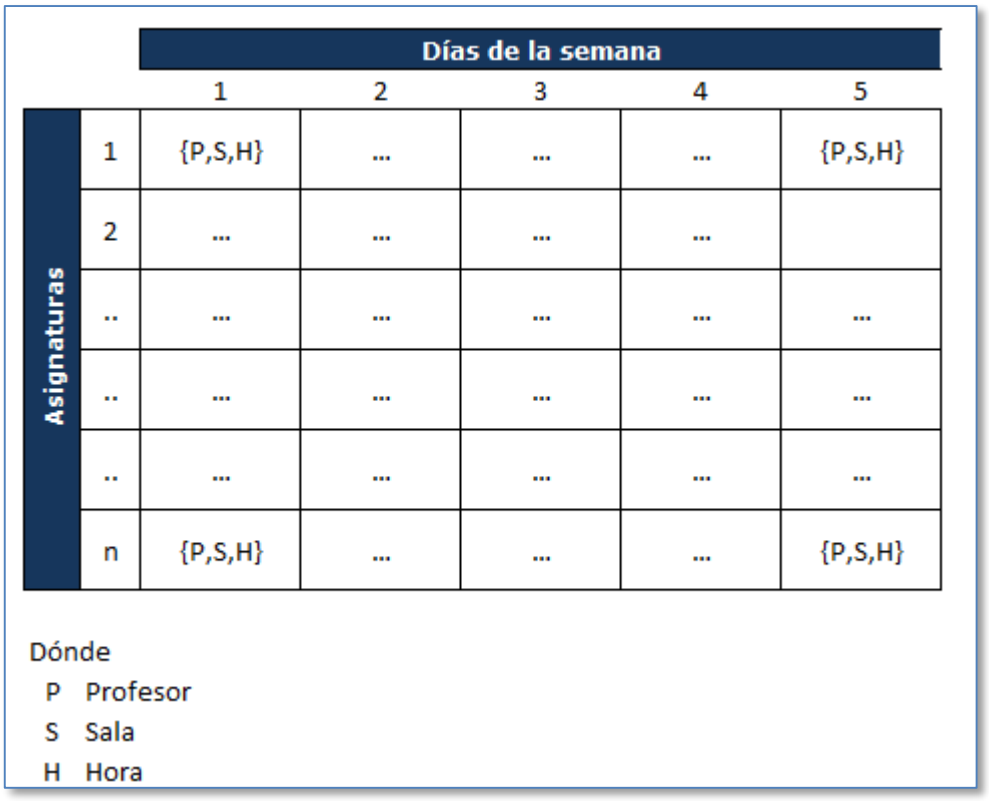


Figura 2: Cromosoma no binario

En un problema de asignación de horarios el cromosoma debe representar las cuatro variables que generalmente se asignan, el evento o clase a realizar, el tiempo en el cual es planificado, el lugar físico y el docente que imparte la clase.

La Figura 3 muestra un ejemplo de cromosoma y genes para el problema de asignación se horarios.

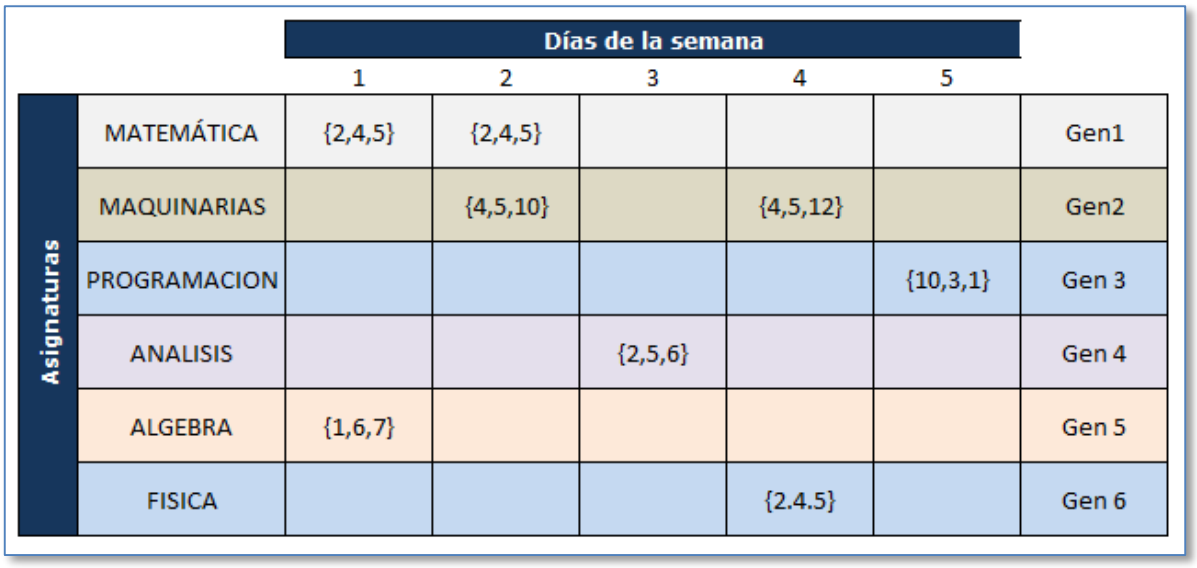


Asignaturas	Días de la semana				
	1	2	3	4	5
1	{P,S,H}	...	...	...	{P,S,H}
2	...	...	...	...	
..	...	...	...	...	...
..	...	...	...	...	...
..	...	...	...	...	...
n	{P,S,H}	...	...	...	{P,S,H}

Dónde
P Profesor
S Sala
H Hora

Figura 3: Representación de un cromosoma para el problema de asignación de horarios.

En la figura anterior se observa que el cromosoma está representado por una matriz de N filas, que corresponden a las asignaturas y cinco columnas que corresponden a los días de la semana (lunes a viernes). En cada celda se encuentran los recursos a asignar, sala, docente y además el tiempo en el cual se realizará una clase.



Asignaturas	Días de la semana 1	2	3	4	5	
MATEMÁTICA	{2,4,5}	{2,4,5}				Gen1
MAQUINARIAS		{4,5,10}		{4,5,12}		Gen2
PROGRAMACION					{10,3,1}	Gen 3
ANALISIS			{2,5,6}			Gen 4
ALGEBRA	{1,6,7}					Gen 5
FISICA				{2.4.5}		Gen 6

Figura 4: Representación de un cromosoma para el problema de asignación de horarios con datos.

Así por ejemplo, la interpretación del gen 1 de la Figura 4 es que la clase de matemática se realiza con el profesor cuyo código es 2, los días lunes y martes, en la sala cuyo código es 4 y en el horario 5. La concatenación de todos los genes corresponde al cromosoma para este problema.

3.3 Función objetivo

La función objetivo, también llamada fitness, es una medida que indica que tan apta es una solución para el problema que se está resolviendo. Corresponde a una medida numérica de la "bondad" de la solución. La función objetivo debe diseñarse de tal modo que encapsule todas las restricciones, fuertes y suaves, del problema. Estas restricciones son representadas con un costo dentro de la función, asignando un costo más alto a las restricciones fuertes y uno más bajo a las restricciones suaves. La aptitud de un cromosoma se obtiene sumando los costos de las restricciones no cumplidas, siendo los cromosomas o soluciones más aptos aquellos que incumplan el menor número de restricciones. Al igual que en el diseño de la representación del individuo para cada problema particular se debe diseñar

una función objetivo. Una función objetivo bien diseñada conduce al algoritmo a seleccionar siempre los mejores individuos para la reproducción.

Se define como convergencia de un algoritmo genético cuando los individuos ya no sufren más cambios y generalmente al lograr esa condición el algoritmo se da por finalizado. Un problema muy común en la ejecución de algoritmos genéticos es la velocidad de convergencia ya que en algunos casos se encuentra una convergencia prematura en la cual el algoritmo converge hacia óptimos locales y en otros casos se produce un efecto contrario, es decir, una convergencia lenta que eleva el tiempo de ejecución del algoritmo. Ambos casos pueden solucionarse mediante transformaciones a la función objetivo.

Se pueden diferenciar cuatro tipos de función objetivo o fitness [Koza, 1992]:

- Fitness Puro: es la medida de ajuste establecida en la terminología natural del propio problema. La ecuación a continuación establece el cálculo del valor de bondad de un individuo i en un instante t (o generación).

$$r(i,t) = \sum_{j=1}^{Nc} |s(i,j) - c(i,j)|$$

Siendo:

s(i,j) = valor deseado para el individuo *i* en el caso *j*

c(i,j) = valor obtenido por el individuo *i* para el caso *j*

Nc = Número de casos

Esta función de fitness mide error por lo cual los individuos con una función objetivo con un valor bajo son los más interesantes.

- Fitness Estandarizado: Para solucionar la dualidad en problemas de minimización o maximización se modifica la función objetivo puro de acuerdo a la ecuación siguiente:

$$s(i,t) = \begin{cases} r(i,t) & minimización \\ r_{max} - r(i,t) & maximización \end{cases}$$

En caso de problemas de minimización se emplea directamente la medida de fitness puro. Si el problema es de maximización se resta de una cota superior r_{max} del error del fitness puro. Empleando esta medida, la bondad de un individuo es mayor cuando más cercano esté a cero el valor del ajuste. Por lo tanto dentro de la generación *t*, un individuo *i* siempre es mejor que uno *j* si se cumple que *s(i,t) < s(j,t).*

- Fitness Ajustado: Se obtiene aplicando la transformación reflejada en la siguiente ecuación al fitness estandarizado:

$$a(i,t) = \frac{1}{1 + s(i,t)}$$

De esta forma, la función objetivo toma siempre valores del intervalo [0.0,1.0]. Cuando más se aproxime la función objetivo de un individuo a 1, mayor será su bondad.

- Fitness Normalizado: Los diferentes tipos de función objetivo vistos en los puntos anteriores hacen referencia únicamente a la bondad del individuo en cuestión. El fitness normalizado introduce un nuevo aspecto: indica la bondad de una solución con respecto al resto de soluciones representadas en la población. Considerando una población de tamaño N, se obtiene la siguiente ecuación:

$$n(i,t) = \frac{a(i,t)}{\sum_{k=1}^{N} a(k,t)}$$

Al igual que el fitness ajustado, siempre toma valores del intervalo [0.0,1.0] y los mejores individuos son aquellos cuya función objetivo se acerque a 1. Pero a diferencia del otro tipo de función objetivo, un valor cercano a uno no solo indica que ese individuo representa una buena solución al problema, sino que además, es una solución destacadamente mejor que las proporcionadas por el resto de la población.

La suma de los valores de la función objetivo normalizada de todos los individuos de una población siempre da 1.

Este tipo de ajuste es empleado en la mayoría de los métodos de selección proporcionales al fitness.

3.4 Estructura general de un algoritmo genético

La estructura general de un algoritmo genético [Spears 1993] puede representarse con los siguientes pasos:

1. Define población inicial.
2. Evalúa población inicial.
3. Selecciona padres.
4. Cruza.
5. Muta.
6. Evalúa.
7. Selecciona y descarta soluciones no aptas.
8. Mientras no exista solución, vuelve a 3.

Dos ejemplos de pseudocódigos de un algoritmo genético pueden visualizarse como sigue a continuación [Pose 2000]:

Inicializar población actual aleatoriamente

MIENTRAS no se cumpla el criterio de finalización

Crear población temporal vacía

MIENTRAS población temporal no llena

Seleccionar padres

Cruzar padres con probabilidad Pc

SI se ha producido el cruce

Mutar uno de los descendientes con probabilidad Pm

Evaluar descendientes

Añadir descendientes a la población temporal

SI NO

Añadir padres a la población temporal

FIN SI

FIN MIENTRAS

Aumentar contador de generaciones

Establecer como nueva población actual la población temporal

FIN MIENTRAS

Figura 5: Pseudocódigo de un algoritmo genético [Pose 2000].

BEGIN ALGORITMO GENÉTICO

Obtener la población inicia al azar

WHILE NOT stop DO

BEGIN

Seleccionar padres de la población

Producir hijos a partir de los padres seleccionados

Mutar los individuos hijos

Extender la población añadiendo hijos

Reducir la población extendida

END

END ALGORITMO GENÉTICO

Figura 6: Pseudocódigo de un algoritmo genético.

3.5 Población inicial

Como se dijo antes, este algoritmo se basa en la selección natural por lo cual debe comenzar con una población inicial que corresponde a posibles soluciones del problema y a partir de ellos construye nuevas soluciones. La población inicial (como también las "futuras" posibles soluciones) es sometida a la evaluación y se seleccionan probabilísticamente aquellos individuos más aptos según la función objetivo. Los cromosomas más aptos son seleccionados como candidatos a reproducirse, es decir dar origen a nuevos cromosomas, o como una solución al problema [Goldberg 1989].

La población inicial puede ser generada de forma aleatoria, mediante una heurística particular, o tomada de una base de conocimiento de alguna solución anterior de otra instancia del problema. En la literatura se observa que al iniciar la población mediante heurísticas, se tiende a tener problemas

de convergencia prematura, es decir, el algoritmo converge hacia soluciones óptimas locales [Goldberg 1989].

Es importante que población inicial tenga un tamaño lo suficientemente grande para garantizar la diversidad de soluciones.

3.6 Operación de selección

Una función de selección se encarga de escoger qué cromosomas son los encargados de reproducirse y cuáles no. Como el algoritmo genético se basa en la naturaleza se otorga mayor oportunidad de reproducción a los individuos más aptos, según su función objetivo, sin embargo no deben descartarse de plano los individuos con una aptitud más deficiente, pues eso le restaría diversidad a las poblaciones. Una técnica común es seleccionar un individuo mediante algún procedimiento formal y otro mediante algún método aleatorio [Pose 2000].

Las técnicas más comunes para seleccionar individuos son selección por ruleta y selección por torneo. La selección por ruleta es probablemente el método de selección más usado en el uso de algoritmos genéticos, consiste en asignar a cada individuo un porcentaje proporcional de la ruleta dependiendo del valor de su función objetivo. Así los individuos más aptos reciben una proporción mayor que los individuos menos aptos. Generalmente los individuos con un porcentaje mayor de la ruleta son ubicados al inicio. Como la suma de la ruleta representa el 100% o el 1, para seleccionar un individuo se obtiene un valor aleatorio entre 0 y 1 y se busca el individuo ubicado en ese valor de la ruleta, recorriendo la ruleta y acumulando las proporciones hasta sobrepasar el valor obtenido.

La selección por ruleta presenta como inconvenientes su ineficiencia a medida que el tamaño de la población aumenta y la probabilidad de que el peor individuo puede ser seleccionado más de una vez.

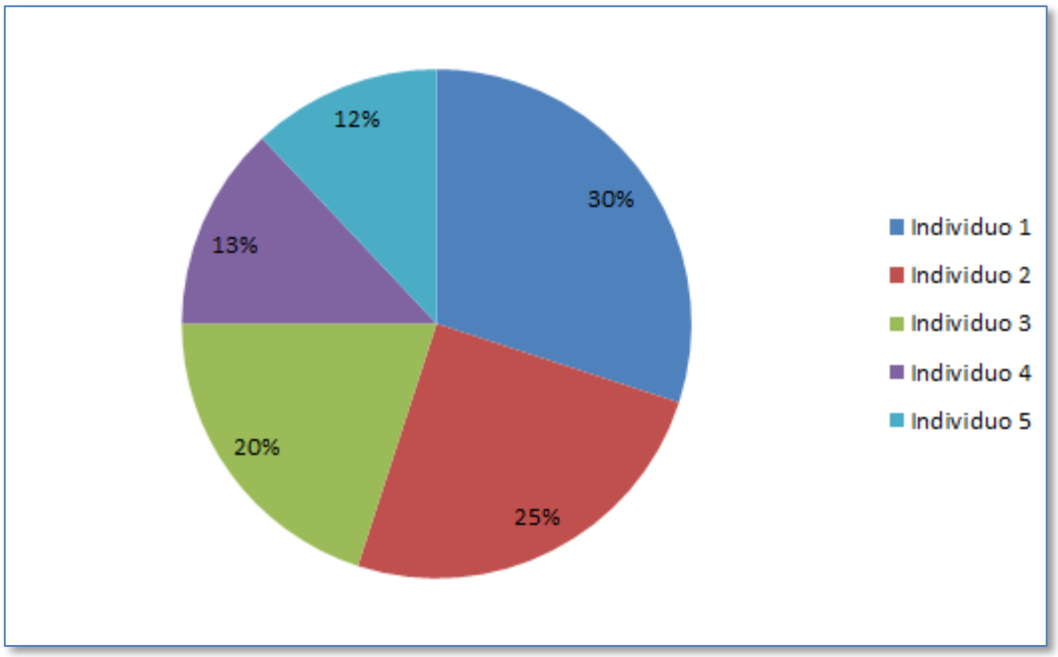


Figura 7: Ejemplo de selección por ruleta.

La selección por torneo consiste en la comparación directa entre individuos. Existen a su vez dos mecanismos de selección por torneo: determinístico y probabilístico. En una selección por torneo determinístico se selecciona un número de individuos al azar, generalmente dos, y se hacen competir, comparando sus aptitudes mediante sus funciones objetivo y el más apto es seleccionado. La selección por torneo en su variante probabilística no siempre selecciona al más apto de la muestra obtenida, ya que aleatoriamente se determina si se eligen al más apto o al menos apto, dando una menor probabilidad de elección a los menos aptos.

Una forma de variar la selección es modificando la cantidad de individuos elegidos para el torneo, así mientras mayor es el número de participante en el torneo, menos opciones de selección tienen los individuos menos aptos. Hay que recordar que no siempre es conveniente desechar los individuos con menores aptitudes ya que su selección permite explorar nuevos espacios de búsqueda.

Existen muchos otros métodos de selección como muestreo determinístico, escalamiento sigma, selección por jerarquías, entre otros.

3.7 Operación de cruza

La cruza es una operación de reproducción sexual. Para buscar nuevas y mejores soluciones, un algoritmo genético utiliza funciones de cruza que corresponde a una recombinación de individuos, dando origen a nuevas generaciones de soluciones. Una función de cruza usa dos cromosomas padres y los combina generando nuevos individuos que a su vez son evaluados y seleccionados como padres de nuevas generaciones [Eiben 2003].

Los diferentes métodos de cruza pueden operar con una estrategia destructiva o no destructiva. La estrategia destructiva consiste en seleccionar como válidos a los nuevos individuos obtenidos de la cruza, a pesar de que sean menos aptos que sus progenitores, mientras que en una estrategia no destructiva solamente prevalecen los individuos cuya medida de bondad es superior a la de los cromosomas que los generaron.

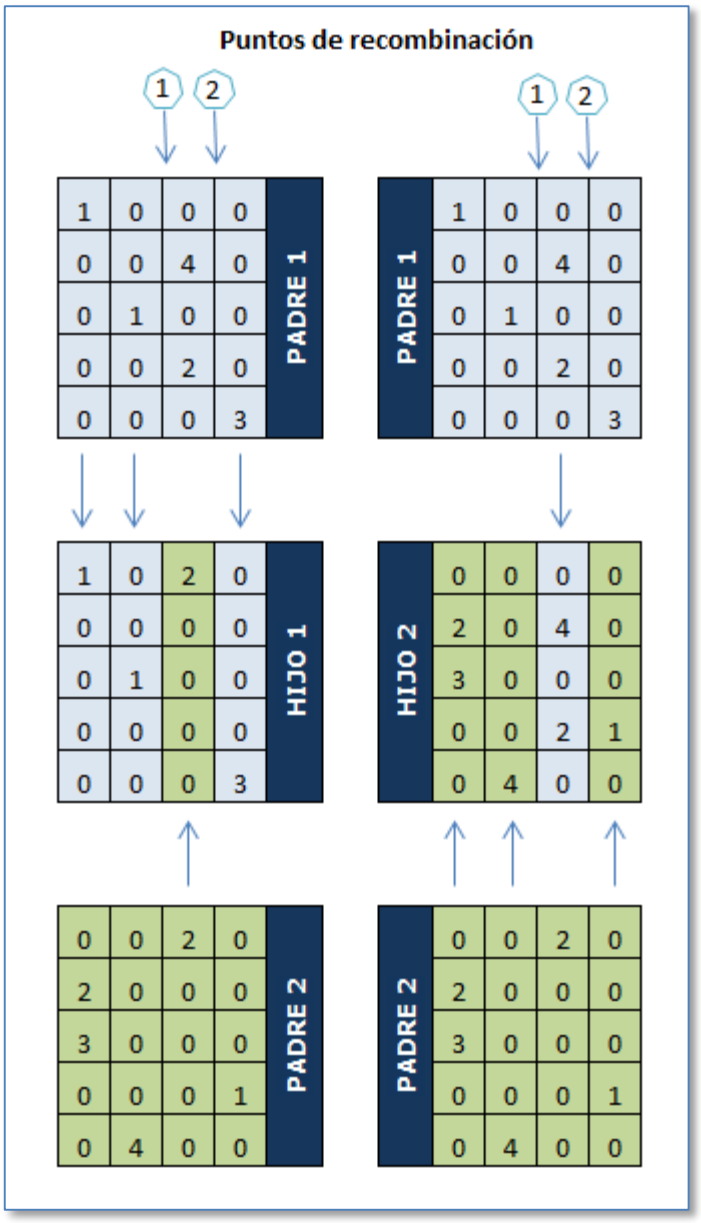


Figura 7: Ejemplo de operación de cruza.

Existen diversas técnicas para la cruza pero las más usadas son las siguientes:

- Cruce de un punto: Es la técnica más sencilla de cruza y consiste en dividir los cromosomas en dos, eligiendo un punto de forma aleatoria. Se debe tener la consideración de que el punto elegido sea un punto intermedio, asegurándose que el cromosoma se divida en dos. Con esta división el cromosoma queda partido en cabeza y cola y se

produce un intercambio de las colas para generar nuevos descendientes. Así, los nuevos individuos conservan información de sus padres.

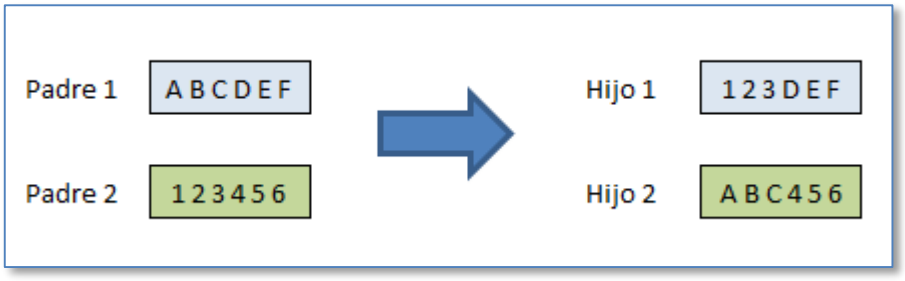


Figura 8: Ejemplo de operación de cruza de un punto.

- Cruce de dos puntos: Es una variante de la cruza de un punto y consiste en dividir los cromosomas en tres, eligiendo dos puntos de forma aleatoria. Se debe tener la consideración de que los puntos elegidos sean puntos intermedio, asegurándose que el cromosoma se divida en tres. Con esta división el cromosoma queda partido en cabeza, segmento central y cola y la descendencia se genera con el segmento central de uno de los padres y los extremos del otro progenitor. Así, los nuevos individuos conservan información de sus padres.

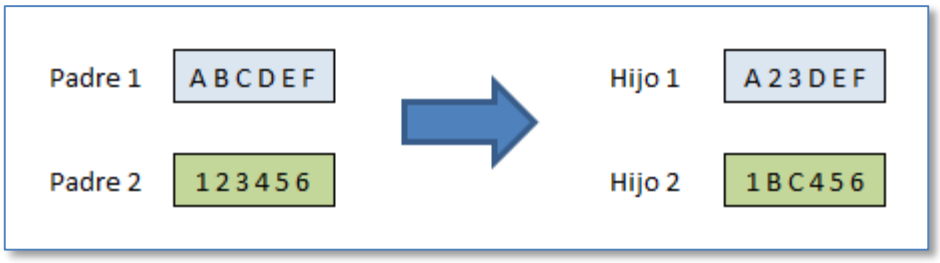


Figura 9: Ejemplo de operación de cruza de un punto.

- Cruce uniforme: Es una técnica en la cual todos los genes de los hijos tiene probabilidades de pertenecer a uno u otro padre. Existen diversas forma de implementarlo, pero generalmente se desarrolla una máscara que decida si el gen se selecciona de un padre u otro.

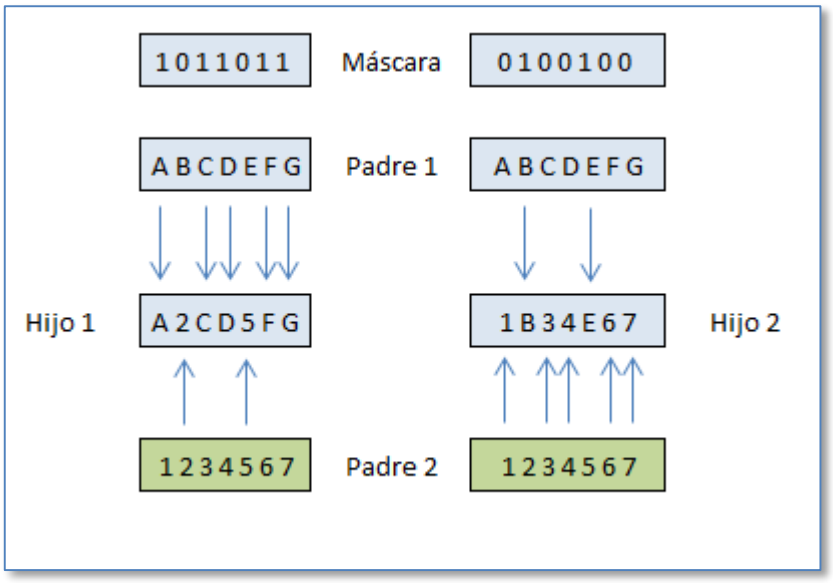


Figura 10: Ejemplo de operación de cruza uniforme.

3.8 Operación de mutación

Una función de mutación se encarga de realizar una modificación en un cromosoma sin combinarlo con otro. La modificación de un individuo consiste en que alguno de sus genes, generalmente solo uno, cambie su valor aleatoriamente. Corresponde al diseño del algoritmo genético decidir si la operación de mutación se realiza seleccionando directamente los individuos y mutarlos o realizar una operación en conjunto con la operación de cruce. Al hacerlo en conjunto con la operación de cruce se seleccionan dos cromosomas para realizar el cruce y si este resulta exitoso se aplica la mutación de uno de los hijos, o ambos. Este proceso tiene su símil en la naturaleza, en la cual se basa el algoritmo genético, ya que en la reproducción el material genético de los descendientes puede tener cierto grado de diferencia o error con respecto al de sus progenitores.

La probabilidad de mutación siempre es muy baja, sin embargo, esto ayuda a explorar nuevas posibles soluciones que no se alcanzarían mediante la cruza [Beligiannis 2008].

Las técnicas de mutación más usadas son las aleatorias, es decir, variar aleatoriamente un gen de un cromosoma. En representaciones binarias de cromosomas esto consiste en obtener la negación de uno de los valores. Otra técnica de mutación muy usada es el intercambio de dos valores de un cromosoma. Un ejemplo de esta última se muestra en la Figura 11.

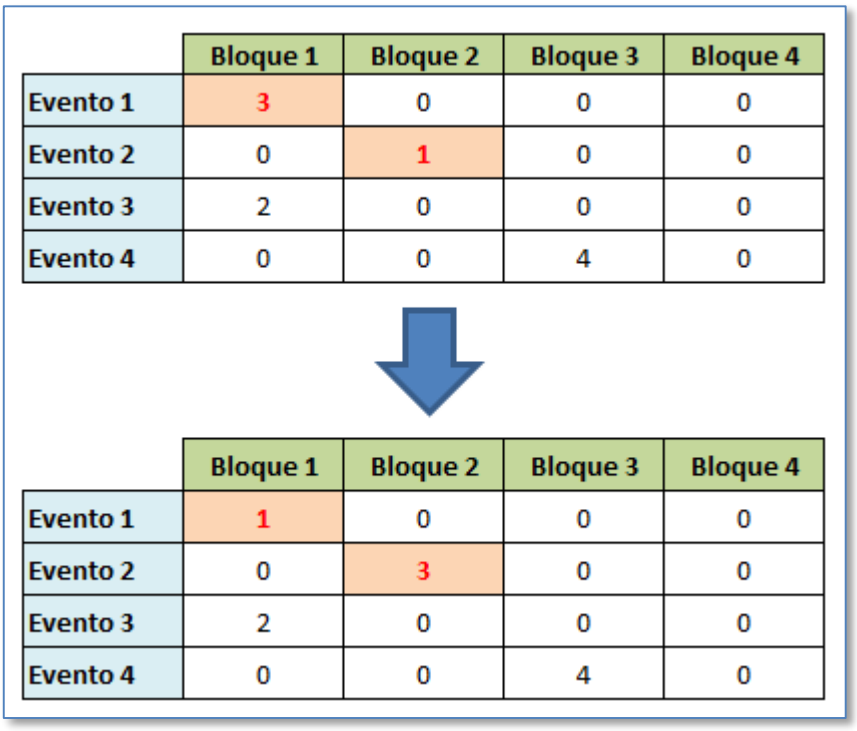


Figura 11: Ejemplo de operación de mutación en una representación para el problema de timetabling.

3.9 Parámetros de un algoritmo genético

Existen algunos parámetros importantes al momento de diseñar un algoritmo genético. Se debe definir el tamaño de población inicial, es decir, con cuantas posibles soluciones comienza el algoritmo. Un número muy pequeño de cromosomas iniciales puede hacer que el algoritmo no cubra de forma adecuada el espacio de búsqueda, mientras que una población inicial extremadamente grande puede agregar costo de procesamiento al algoritmo.

Otra de las decisiones que se debe tomar al diseñar un algoritmo genético es la condición de término de éste. Las condiciones de término pueden estar definidas por el tiempo, es decir, fijar a priori un tiempo de ejecución del algoritmo y una vez cumplido se selecciona la mejor solución que se tenga. Por otro lado, también se puede definir una cantidad de iteraciones para el algoritmo, lo cual coincide con la cantidad de generaciones que se desea explorar.

Lo más usual es que la condición de término del algoritmo esté dada por la convergencia de este, es decir, cuando los cambios entre una generación y sus descendientes dejen de ser significativos.

3.10 Variantes de algoritmos genéticos

Existen diversas variantes o hibridaciones para los algoritmos genéticos, entre los que se puede mencionar:

- Algoritmos meméticos: Corresponde a un algoritmo genético en su estructura general, pero su principal característica es que se mantiene constante el tamaño de la población o posibles soluciones. En cada iteración del algoritmo se reemplaza una alternativa de la población mediante algún método de modificación.
- Búsqueda Tabú: Es una meta heurística basada en búsqueda local. Es un mecanismo que busca soluciones en vecindades y al encontrar alguna potencial solución, establece sobre ella una marca o "tabú" de tal modo de no volver a visitarla.
- Colonia de hormigas: Corresponde a una técnica que se inspira en el comportamiento real de las hormigas. Es una meta heurística.
- Coloración de grafos.

Se encuentra en la literatura el uso de combinaciones de algoritmos genéticos para obtener mejores soluciones [Hernández 2008]

4. METODOLOGÍA Y DESARROLLO

4.1 Justificación y contexto

INACAP es la institución de educación superior más grande de Chile en cuanto a cantidad de alumnos e infraestructura. Cuenta con un sistema de educación integrado cuyo pilar fundamental es la continuidad de estudios que el alumno puede realizar entre el Centro de Formación Técnica INACAP (CFT), el Instituto Profesional INACAP (IP) y la Universidad Tecnológica de Chile (UTC). Este modelo ha llevado a INACAP a tener actualmente cerca de 120.000 alumnos, repartidos en sus 26 sedes, que lo hacen tener presencia en todas las regiones del país. La institución ha duplicado su cantidad de alumnos en los últimos 10 años, por lo cual todos los procesos que se podían realizar de forma manual se complejizan por el volumen de información que manejan a pesar de que para apoyar toda su gestión académica INACAP cuenta con una moderna plataforma tecnológica denominada Sistema Integrado de Gestión Académica (SIGA).

La creación de horarios de forma manual es uno de los procesos que más problemas presenta en la actualidad debido a la complejidad intrínseca que posee. Si bien cada período académico se llega a contar con un horario de clases, el costo de realizarlo es demasiado alto.

En el contexto de la generación de horarios académicos, el SIGA permite administrar toda la información de entrada para el proceso, la generación de cursos, la recolección de datos de docentes, la administración de las salas y la proyección de cantidades de alumnos para períodos futuros. Sin embargo, el SIGA no entrega ninguna propuesta de horario y solo provee una interfaz para que el usuario pueda realizar la planificación.

INACAP en sus 26 sedes a nivel nacional dicta carreras de sus tres instituciones (CFT, IP y UTC) en un mismo espacio físico. Las carreras son segmentadas según áreas del conocimiento y cada área esta comandada por un Director de Carrera quien es el responsable de realizar la creación de los

horarios para sus alumnos. No se tiene una forma única de planificar los cursos y el uso de la salas ya que existen métodos que dependen del tamaño de la sede. Algunas sedes dividen las salas asignando a cada Director de Carrera un grupo de salas. Otras sedes priorizan y asignan orden de planificación para los Directores de Carreras, teniendo los primeros mayores opciones de usar las mejores salas y el mejor horario. Mientras que otras sedes realizan una asignación conjunta de todas sus carreras paralelamente.

La planificación del horario se realiza sin asignar a los alumnos. Estos deben posteriormente elegir las secciones (cursos) que más les acomoden.

Los alumnos al momento de matricularse eligen pertenecer a una jornada: diurna o vespertina, lo cual le asegura que sus clases se realizan en el horario seleccionado por él.

Algunos de los datos relevantes para este proceso son los siguientes:

- Cada año se presupuestan y se crean, previo a la planificación, las secciones (cursos) que se dejan disponibles a los alumnos para su proceso de toma de carga académica. Una sección tiene los siguientes datos relevantes:
    - Asignatura.
    - Carrera.
    - Jornada (diurna o vespertina).
    - Nivel de la malla curricular al cual está asociado (de acuerdo a la carrera).
    - Cantidad de horas semestrales.
    - Cantidad máxima de alumnos.
    - Cantidad de días por semana en que se pueden hacer las clases.
    - Prioridad para ser planificada en los primeros bloques del día.

- Existe un cuerpo docente para los cuales se tiene la siguiente información:
    - Asignaturas que pueden dictar.
    - Disponibilidad horaria.
    - Sede en la cual puede hacer clases.
    - Máximo de horas.
    - Prioridad para ser asignado a una asignatura en desmedro de otros docentes.
- Cada sede cuenta con salas en las cuales se realizan las clases. Estas salas están clasificadas según:
    - Tipo de sala (sala, laboratorio, gimnasio, etc.).
    - Capacidad máxima.
    - Prioridad según el estado en que se encuentra la sala o el mobiliario.
    - Carreras que alberga: Algunas sedes tienen repartidas sus salas según carrera, otras asignan libremente.
    - Ubicación.

- Cada sede cuenta con un horario definido. Generalmente de 18 o 19 bloques diarios. Estos bloques están clasificados según jornada: diurna y vespertina.

La generación de horarios en INACAP se debe realizar por sede y debe considerar las siguientes restricciones:

Restricciones duras:

- Un docente no puede asignarse a dos o más cursos en un mismo horario.
- No se pueden planificar dos o más cursos de una misma carrera y un mismo nivel en un mismo horario.
- No se pueden planificar dos o más cursos en una misma sala en un mismo horario.
- A un docente no se le puede planificar más horas que el máximo de horas diarias y semanales.
- Todo curso debe tener asignado una sala y un docente.
- Las secciones debe programarse en horario según su jornada, diurna o vespertina.
- No se puede asignar una sección en una sala que tenga un cupo inferior a la cantidad de alumnos de la sección.

Restricciones suaves:

- Se debe minimizar las "ventanas" para los docentes.
- Se debe minimizar las "ventanas" en la planificación de cursos de un mismo nivel y carrera.
- Se debe asignar en primer lugar a los profesores con mayor prioridad.
- Se debe asignar en primer lugar las salas con mayor prioridad.

Estos antecedentes sustentan este proyecto, ya que una herramienta que ofrezca alternativas de horarios académicos en un tiempo menor, libera recursos que permiten realizar una mejor gestión.

4.2 Descripción del problema

La elaboración de un horario académico consiste en asignar para cada curso un docente y un lugar físico, o sala, en el cual realizar cada una de las

clases. Para entender el proceso de generación de horarios académicos en INACAP es necesario mencionar algunos términos y conceptos generales.

INACAP cuenta con 26 sedes a lo largo de Chile, en cada una de las cuales dicta carreras de sus tres instituciones: Centro de Formación Técnica, Instituto Profesional y Universidad. Los alumnos al momento de matricularse en una carrera deben elegir si desean asistir a clases en jornada diurna o vespertina. Por reglamento académico a un alumno se le debe respetar su horario según la jornada escogida.

Cada carrera cuenta con una *malla curricular* que define las asignaturas que los alumnos de la carrera deben cursar. Según esto, cada asignatura está asociada a un nivel o semestre en el cual a cada alumno le corresponde cursar la asignatura.

INACAP divide el año académico en cuatro períodos o semestres: verano, primavera, invierno y otoño. Los semestres principales y en los cuales se efectúan matrículas de alumnos son otoño (primer semestre) y primavera (segundo semestre), mientras que los semestres de invierno y verano, también llamados inter-semestres, solo se utilizan para cursar asignaturas rezagadas. La asignación de horarios aborda solo los semestres de otoño y primavera, ya que las asignaturas que se ofrecen en inter-semestre son muy pocas.

Los horarios de las clases son de lunes a viernes y se dividen en 18 bloques horarios de 45 minutos cada uno, con distintos horarios de inicio y término según la sede y con distintos horarios de receso por recreos u horarios de colación o almuerzo. En cada sede los bloques horarios son clasificados según jornada. Aunque existe un estándar institucional, la diferencia de hora puede hacer variar esa clasificación. Así por ejemplo, en una sede de Santiago los bloques desde el 1 al 12 (ordenados según la hora de inicio) corresponden a jornada diurna, mientras que los bloques 13 al 18 corresponden a jornada vespertina. Pero en una sede del sur como

Coyhaique, donde las clases comienzan más tarde, la jornada diurna puede abarcar los bloques 1 al 13 o incluso 14, dejando menos bloques horarios a la jornada vespertina.

Cada semestre se planifica, en base a la proyección de matrícula de alumnos, cuántas *secciones* o cursos son necesarios para cada asignatura de cada carrera. Se planifican secciones para cada jornada, dependiendo de si existe oferta de matrícula en la jornada respectiva. Para cada sección se define el número máximo de alumnos que puede contener, las horas semanales que se deben realizar (según el programa de cada asignatura) y también la cantidad máxima de horas diarias que se pueden planificar. Opcionalmente se puede definir que se requiere que las clases de esa sección se realicen en días continuos o con por lo menos un día de separación. También se debe definir excepcionalmente si la sección requiere de algún espacio físico distinto a una sala tradicional, como un taller, un laboratorio, un gimnasio o si se realiza en terreno, es decir, fuera de las dependencias de la institución. También es un dato opcional de la sección el horario preferido para realizar su clase, por ejemplo cierta asignatura puede tener mejores resultados si las clases son a primera hora de la mañana o alguna asignatura complicada puede definir que para evitar el ausentismo se planifiquen sus clases en horarios de tarde.

La infraestructura se encuentra clasificada y tabulada. Así cada sede cuenta con salas, talleres, laboratorios, gimnasios y terrenos. En adelante llamaremos sala a cualquiera de estos espacios. Cada sala tiene definida su capacidad y su capacidad máxima. La capacidad indica cuantos alumnos se podrían ubicar en ese recinto actualmente y la capacidad máxima indica cual es el cupo máximo que se podría llegar agregando más mobiliario a esa sala. Para efectos de este problema es relevante solo la capacidad. La capacidad no es relevante cuando el espacio a asignar es un gimnasio o terreno, ya que pueden albergar alumnos de forma "ilimitada". Para todos los espacios se

administra su vigencia, es decir, algunas salas podrían no ser ocupadas por no encontrarse en condiciones óptimas. Opcionalmente algunas salas tienen preferencia de uso, o sea, las que se encuentran en mejores condiciones, se espera que sean más usadas que aquellas más desmejoradas.

Los docentes semestre a semestre deben completar un registro en el cual indican su disponibilidad horaria para realizar clases y las asignaturas que desean impartir. Esta información es validada por un administrativo que corrobora que el docente cumple con el perfil para dictar las asignaturas seleccionadas. Opcionalmente un docente puede indicar cuál es su horario de preferencia para dictar sus clases.

4.3 Organización y representación de los datos de entrada

Para el diseño de un algoritmo genético es muy importante la representación de los datos, no solo de la solución o de los cromosomas, sino también de los datos de entradas. Si bien estos ya se encuentran en bases de datos relacionales, para hacer óptima su recuperación y uso se definen estructuras matriciales que deben ser llenadas antes de ejecutar el algoritmo genético. Estas estructuras corresponden a los input principales del algoritmo genético.

A continuación se describen las estructuras de datos para la información de entrada.

4.3.1 Secciones

Se define el conjunto de secciones para una sede de INACAP y *n* la cantidad de secciones a dictar en una sede.

La representación de las secciones es una matriz de *n* filas, donde cada columna representa un dato relevante para el problema, tal como se muestra en la Figura 12.

MATRIZ DE SECCIONES									
ID	CODIGO	ASIGNATURA	SEDE	CARRERA	NIVEL	JORNADA	HORAS SEMESTRALES	HORAS SEMANALES	ALUMNOS
1	14105091	SB0206	34	48	4	1	50	4	40
2	14105092	SB0206	34	48	4	1	60	6	40
3	14105093	SB0207	34	48	4	1	36	2	40
4	14105095	SB0207	34	48	4	1	36	2	40
..	..	..	..	..	..	..	..	..	..
n	14105801	SC0401	34	48	2	1	36	2	35

Figura 12: Representación de las secciones.

Dónde:

Id: identificador único en la matriz. Sirve para identificar una sección dentro del algoritmo genético.

Código: Identificador único de la sección en el Sistema Integrado de Gestión Académica SIGA.

Asignatura: Indica el código de la asignatura a la cual pertenece la sección. Se utiliza para asignar docentes que correspondan a esa asignatura.

Sede: Indica el código de la sede a la cual pertenece la sección.

Carrera: Código de carrera a la cual pertenece una sección. Se utiliza para evaluar la restricción de tope horario.

Nivel: Nivel o semestre al cual pertenece la asignatura en la malla curricular de la carrera. Se utiliza para evaluar la restricción de tope horario.

Jornada: Código de jornada de la sección. Se utiliza para evaluar la restricción de horarios según jornada.

- Código 1 indica jornada diurna.
- Código 2 indica jornada vespertina.

Horas Semestrales: Valor que indica la cantidad de horas semestrales de clases que debe cumplir la sección. Se utiliza para asignar el horario a la sección.

Horas Semanales: Valor que indica la cantidad de horas semanales de clases que debe cumplir la sección. Se utiliza para asignar el horario a la sección. Se calcula a partir de las horas semestrales considerando que un semestre estándar tiene dieciocho semanas de clases.

Alumnos: Valor que indica la cantidad máxima de alumnos que acepta la sección. Se utiliza para evaluar la restricción de capacidad de sala versus alumnos.

Adicionalmente se define para cada sección la cantidad de bloques horarios que se deben asignar para cada día, tal como se muestra en la figura siguiente:

MATRIZ DE BLOQUES POR SECCIÓN					
ID	BLOQUES DIA1	BLOQUES DIA 2	BLOQUES DIA3	BLOQUES DIA4	BLOQUES DIA5
1	2	2	0	0	0
2	2	2	2	0	0
3	3	0	0	0	0
4	3	3	0	0	0
..	..	..	..	..	..
n	2	0	0	0	0

Figura 13: Representación de la programación semanal de las secciones.

En la Figura 13 se puede ver por ejemplo que para la sección cuyo ID es 1 se indica que se debe asignar planificación en dos días de la semana (independientemente de cuáles sean esos días) y que en cada día se debe planificar dos bloques de clases. Del mismo modo para la sección cuyo ID es 3 en el ejemplo se debe asignar planificación solo un día de la semana, con tres bloques de clases. Se debe considerar que siempre los bloques horarios deben ser continuos.

Las restricciones asociadas a secciones son las siguientes:

- A cada sección se le debe asignar un docente que pueda impartir la asignatura.

- A cada sección se le debe asignar una sala con un cupo igual o mayor a la cantidad máxima de alumnos de la sección.
- A cada sección se le debe asignar una sala en la cual se realiza cada clase.
- No se puede asignar en un mismo bloque horario dos secciones que pertenezcan a la misma carrera y al mismo nivel.

Para cumplir con ésta última restricción de evitar que se asignen en un mismo bloque horario secciones que pertenezcan a la misma carrera y nivel es que se define la estructura de curso. Esta estructura permite dar libertad a esta restricción, logrando por ejemplo evitar choques horarios entre asignaturas de distintos niveles, pero que por tener altos porcentajes de reprobación no se recomienda que se planifiquen paralelamente con asignaturas de otro nivel.

La estructura de curso se muestra en la Figura 14.

MATRIZ DE CURSOS		
ID SECCIÓN	ID CURSO	NOMBRE CURSO
1	1	PRIMER SEMESTRE - INGENIERIA
2	1	PRIMER SEMESTRE - INGENIERIA
3	1	PRIMER SEMESTRE - INGENIERIA
1	2	SEGUNDO SEMESTRE - INGENIERIA
4	2	SEGUNDO SEMESTRE - INGENIERIA
5	3	TERCER SEMESTRE - INGENIERIA
6	3	TERCER SEMESTRE - INGENIERIA
..	..	..
n	c	CURSO C

Figura 14: Representación de cursos.

De la tabla anterior se interpreta por ejemplo que la sección cuyo ID es 1, no puede planificarse en paralelo con ninguna clase de las secciones cuyo ID es 2 y 3, por pertenecer al mismo semestre académico. Además, la misma sección 1 no se debe planificar en paralelo con la sección cuyo ID es 4, que

pertenece a un nivel o semestre más avanzado, debido a que la asignatura de la sección 1 podría tener un alto índice de reprobación.

4.3.2 Salas

Se define el conjunto de salas para una sede de INACAP y la representación del conjunto de salas es una matriz de *i* filas, donde cada columna representa un dato relevante para el problema, como se muestra en la siguiente figura:

MATRIZ DE SALAS				
ID	CÓDIGO	CAPACIDAD	PRIORIDAD	TIPO
1	14567	20	0	1
2	23467	25	1	2
3	32367	15	2	3
4	41267	0	0	4
5	50167	25	1	1
6	59067	50	2	2
7	67967	100	0	3
..	..	..	..	..
i	87999	20	0	1

Figura 15: Representación de las salas.

Dónde:

Id: identificador único en la matriz. Sirve para identificar una sala dentro del algoritmo genético.

Código: Identificador único de la sala en el Sistema Integrado de Gestión Académica SIGA.

Capacidad: Valor que indica la cantidad máxima de alumnos que alberga la sala. Valor 0 indica que no tiene límite de alumnos (gimnasios y terrenos). Se utiliza para validar la restricción de cupos de alumnos en la sala.

Prioridad: Código de prioridad de la sala. Se utiliza para evaluar la restricción de asignación según prioridad.

- Código 0 indica que no existe prioridad para la sala.

- Código 1 indica que de preferencia se debe usar la sala.
- Código 2 indica que de preferencia no se debe usar la sala.

Tipo: Código que indica el tipo de sala.

- Valor 1 indica sala.
- Valor 2 indica laboratorio.
- Valor 3 indica taller.
- Valor 4 indica gimnasio.
- Valor 5 indica terreno.

Las últimas tipificaciones de Prioridad y Tipo se resumen en una estructura que indica para cada sección que tipo de sala se puede usar.

Las restricciones asociadas a salas son las siguientes:

- A cada sección se le debe asignar una sala con un cupo igual o mayor a la cantidad máxima de alumnos de la sección.
- No se puede asignar más de una sección a una sala en el mismo bloque horario.

4.3.3 Docentes

Se define el conjunto de docentes para una sede de INACAP y j la cantidad de docentes de una sede.

La representación de los docentes es una matriz de j filas, donde cada columna representa un dato relevante para el problema, como se muestra en la Figura 16.

MATRIZ DE DOCENTES				
ID	CÓDIGO	MÁXIMO SEMANA	MÁXIMO DIA	PRIORIDAD
1	56784	45	9	0
2	84256	45	9	1
3	111728	45	8	2
4	139200	45	8	0
5	166672	45	9	1
6	194144	45	9	2
7	221616	45	9	0
..	..	..	..	..
i	276560	35	7	1

Figura 16: Representación de los docentes.

Dónde:

Id: identificador único en la matriz. Sirve para identificar un docente dentro del algoritmo genético.

Código: Identificador único del docente en el Sistema Integrado de Gestión Académica SIGA.

Máximo Día: Valor que indica la cantidad máxima de horas diarias que un docente puede realizar. Se utiliza para validar la restricción de horas diarias de un docente.

Máximo Semana: Valor que indica la cantidad máxima de horas semanales que un docente puede realizar. Se utiliza para validar la restricción de horas semanales de un docente.

Prioridad: Código de prioridad del docente. Se utiliza para evaluar la restricción asignación de docentes según prioridad.

- Código 0 indica que no existe prioridad para el docente.
- Código 1 indica que de preferencia se debe usar el docente.
- Código 2 indica que de preferencia no se debe asignar el docente.

Las restricciones asociadas a docentes son las siguientes:

- No se puede asignar un docente a más de un curso en el mismo horario.
- No se puede asignar a un docente más horas que el máximo diario permitido.
- No se puede asignar a un docente más horas que el máximo semanal permitido.

4.3.4 Horarios

Se define el conjunto de bloques horarios para una sede de INACAP. La cantidad de bloques puede variar, ya que existen sedes que planifican 18 bloques diarios repartidos en cinco días de la semana y otras sedes que planifican 19 bloques diarios.

La representación de los bloques de horarios para una sede de 18 bloques diarios es una matriz de 90 filas como se muestra en la Figura 17.

MATRIZ DE HORARIOS		
ID	DIA	BLOQUE
1	LUNES	1
2	LUNES	2
3	LUNES	3
4	LUNES	4
5	LUNES	5
6	LUNES	6
7	LUNES	7
..	..	..
90	VIERNES	18

Figura 17: Representación de los bloques horarios.

4.4 Organización y representación de estructuras de validación

Además de las estructuras que almacenan los datos de entrada, se definen una serie de estructuras usadas para ser consultadas por la función objetivo. Estas estructuras precargan algunas restricciones que deben ser cumplidas.

A continuación se describen las estructuras de datos para la validación.

4.4.1 Combinación docentes - horario

Se define el conjunto de combinaciones válidas entre docentes y horarios. Esta estructura se inicia basándose en la disponibilidad horaria entregada por el docente y validada por el Director de Carrera.

La representación de las combinaciones válidas de docentes y horarios es una tabla, donde cada fila representa el docente y un horario (día y bloque horario) y un indicador de habilitado o no habilitado (ver Figura 18).

MATRIZ DE DISPONIBILIDAD DOCENTE			
ID DOCENTE	DIA	HORA	HABILITADO
1	1	4	1
1	1	5	1
3	4	7	1
3	4	8	1
3	4	9	1
5	2	1	0
5	2	2	1
..	..	..	..
i	1	1	1

Figura 18: Representación de las combinaciones válidas de docentes y horarios.

En la matriz de combinaciones válidas entre docentes y horarios cada fila pertenece a un docente, un día y un bloque horario y un flag donde valor 0 indica que el docente no puede hacer la clase en el bloque. El valor 1 indica que el docente si puede impartir clase en el bloque. En la Figura 18 se aprecia por ejemplo que el docente cuyo id es 1 puede hacer clases día lunes (día 1) y en los bloques horarios 4 y 5, mientras que el docente cuyo id es 5 no puede impartir clase el día martes (día 2) en bloque 1.

A partir de esta disponibilidad docente se genera una estructura de datos con el resumen de disponibilidad para cada docente, que se muestra en la Figura 19.

MATRIZ DE RESUMEN DISPONIBILIDAD DOCENTE			
ID DOCENTE	DIAS DISPONIBLES	HORAS DIURNAS DISPONIBLES	HORAS VESPERTINAS DISPONIBLES
1	2	14	3
2	3	8	4
3	1	10	5
4	4	8	2
5	5	9	3
6	3	3	4
7	2	2	0
..	..	..	..
j	2	5	6

Figura 19: Representación del resumen de disponibilidad docente.

En esta estructura se resumen para cada docente lo siguiente:

Días disponibles: Cantidad de días semanales en que un docente tiene disponibilidad para impartir clases.

Horas Diurnas Disponibles: Cantidad de horas semanales en horario diurno disponibles para impartir clases por un docente.

Horas Vespertinas Disponibles: Cantidad de horas semanales en horario vespertino disponibles para impartir clases por un docente.

Todos estos datos se utilizan para depurar la asignación de docentes a una sección.

4.4.2 Combinación docentes - secciones

Se define el conjunto de combinaciones válidas entre docentes y asignaturas. Esta estructura se inicia basándose en la disponibilidad entregada por el docente y validada por el Director de Carrera.

La representación de las combinaciones válidas de docentes y secciones es una matriz donde cada fila representa una combinación docente- sección, la cual se muestra en la Figura 20.

MATRIZ DE COMBINACIÓN DOCENTE-SECCIÓN		
ID DOCENTE	ID SECCION	HABILITADO
1	1	1
1	2	1
2	3	1
2	1	1
2	3	1
3	1	0
3	4	0
..	..	..
j	n	1

Figura 20: Representación de las combinaciones válidas de docentes y secciones.

En la matriz de combinaciones válidas entre docentes y secciones cada fila corresponde a una combinación de un docente con una sección y un indicador o flag de validez. El valor 0 en el flag indica que el docente no puede impartir la sección. El valor 1 indica que el docente si puede impartir la sección. En la figura 20 se aprecia que el docente cuyo id es 1 puede impartir la sección cuyo id es 1 y 2, mientras que el docente cuyo id es 3 no puede impartir la sección cuyo id es 1 y 4.

En esta estructura se encuentran solo las combinaciones entre docentes y secciones que han sido indicadas como válidas, es decir, docentes que por su formación puedan impartir la asignatura que corresponde a la sección. Inicialmente se generan todas las combinaciones según el criterio antes indicado, sin filtrar por disponibilidad horaria, es decir todas las combinaciones existentes en esa estructura estarían en condición de habilitadas.

Estos datos son depurados antes de ser ingresados al algoritmo, basándose en la representación de la programación semanal de las secciones (Figura 13), de la que se puede deducir cuantos días a la semana se requieren clases y cuantos bloques horarios totales requiere en la semana, y en el resumen de disponibilidad docente (Figura 19) en el cual se indica cuantos días

disponibles tiene un docente en una semana y cuantos bloques horarios por cada jornada puede impartir clases. Al cruzar esta información se descartan aquellos docentes cuya disponibilidad no cubre las necesidades horarias de la sección, ya sea por cantidad de días o por cantidad de bloques horarios según la jornada.

4.4.3 Combinación secciones - salas

Se define el conjunto de combinaciones válidas entre secciones y salas. Esta estructura se inicia basándose en la cantidad de alumnos de una sección y el cupo de una sala además del tipo de sala que requiere una sección, lo cual es planificado por un Director de carrera.

La representación de las combinaciones válidas de secciones y salas es una tabla donde cada fila representa una combinación entre la sección y una sala indicando si la sala se encuentra habilitada para acoger a los alumnos de la sección.

MATRIZ DE COMBINACIÓN SECCIÓN-SALA		
ID SECCIÓN	ID SALA	HABILITADO
1	20	1
1	21	1
1	22	1
1	23	1
1	24	1
2	31	0
2	32	1
..	..	..
j	i	1

Figura 21: Representación de las combinaciones válidas de secciones y salas.

En la matriz de combinaciones válidas entre secciones y salas, cada fila corresponde a una combinación entre sección y sala con un indicador de habilitado o deshabilitado. El valor 0 indica que la sección no puede realizarse en la sala. El valor 1 indica que las clases de la sección si pueden realizarse en la sala. En la Figura 21 se aprecia por ejemplo que la sección

cuyo id es 1 puede hacer clases en las salas cuyo id es 20, 21, 22, 23 y 24 mientras que la sección cuyo id es 2 no puede impartir clase en la sala cuyo id es 31. Esta estructura encapsula las restricciones de capacidad de la sala y tipo de sala.

4.4.4 Disponibilidad horaria de salas

Se define el conjunto de combinaciones válidas entre salas y horarios. Esta estructura se inicia basándose en una planificación ya realizada restringiendo el uso de las salas que ya se encuentran utilizadas. También permite asignaciones parciales de horarios.

La representación de las combinaciones válidas de salas y horarios es una tabla donde cada fila representa una combinación entre la sala y un horario indicando si la sala se encuentra habilitada para ser utilizada en ese horario.

MATRIZ DE DISPONIBILIDAD HORARIA DE SALAS			
ID SALA	DIA	HORA	HABILITADO
1	1	1	1
1	2	1	1
1	3	1	1
1	4	1	1
1	5	1	1
2	1	1	0
2	1	2	0
..		..	..
j	5	18	1

Figura 22: Representación de la disponibilidad horaria de las salas.

En la matriz de combinaciones válidas entre salas y horarios, cada fila corresponde a una combinación entre sala y horario con un indicador de habilitado o deshabilitado. El valor 0 indica que la sala no puede usarse en el horario referido. El valor 1 indica que sala si puede utilizarse en el horario referido. En la Figura 22 se aprecia por ejemplo que la sala cuyo id es 1 puede usarse los días lunes a viernes (1 al 5) en el primer bloque horario

mientras que la sala cuyo id es 2 no puede usarse los días lunes en el bloque 1 y 2.

4.4.5 Secciones asignables

Basándose en la disponibilidad de docentes y salas para una sección se establece el universo de secciones que se tratan de planificar en el algoritmo genético. Para esto se resume la información anterior en la estructura que se muestra en la Figura 23.

MATRIZ DE ASIGNACIÓN DE SECCIONES		
ID	DOCENTES DISPONIBLES	SALAS DISPONIBLES
1	2	2
2	2	2
3	3	0
4	3	3
..	..	..
n	2	0

Figura 23: Representación de las secciones asignables.

En la matriz de la figura anterior se resume la información de disponibilidad de una sección. Por ejemplo en la Figura 23 se visualiza que la sección cuyo id es 1 tiene dos docentes disponibles para asignar a su planificación y que se puede planificar en dos salas, mientras que la sección cuyo id es 3 no tiene salas disponibles para planificar. El algoritmo genético solo considera como "*asignables*" aquellas secciones que tiene al menos un docente disponible y una sala disponible, el resto de las secciones queda sin planificación.

4.5 Estructura de solución o cromosoma

La codificación del cromosoma es uno de los factores más influyentes en el rendimiento de un algoritmo genético.

En el diseño de este algoritmo se opta por un cromosoma no binario, es decir con que codifican directamente cada parámetro con valores enteros o reales ya que eso permite una mejor comprensión del problema.

Inicialmente se opta por un diseño de cromosoma en columnas que permitiera manejar un cromosoma con una cantidad limitada de filas y columnas. Las filas estaban limitadas a la cantidad de secciones y cada columna representaba uno de los valores a asignar a esa sección como el docente y la sala, mientras que el horario se manejaba como un vector binario de 90 posiciones en que cada posición se codificaba con un valor binario que indicaba que esa sección se planificaba en el horario indicado.

Esta representación inicial se muestra en la Figura 24.

CROMOSOMA										
ID SECCIÓN	ID DOCENTE	ID SALA	B01	B02	B03	B04	B05	B06	..	B90
1	7	23	1	1	0	0	0	0	..	0
2	5	2	0	0	1	1	1	0	..	0
3	12	23	1	0	0	1	1	0	..	1
4	3	3	1	1	1	0	0	0	..	0
..	..	..	..	..	..	..	..	..	..	..
n	6	45	0	0	0	0	0	0	..	0

Figura 24: Representación inicial del cromosoma.

En la figura anterior se representa una posible solución del problema con la representación inicial del cromosoma. Este diseño permite disminuir la cantidad de filas que debe manejar el algoritmo y permite de manera fácil medir por ejemplo la cantidad de choques horarios entre docentes o una misma sala. Por ejemplo en la figura anterior se ve claramente que la sala cuyo id es 23 tiene una doble asignación en el primer bloque. Esto permite que las restricciones más elementales sean fáciles de calcular. Sin embargo, las operaciones de cruza y mutación se hacen difíciles de manejar debido al gran tamaño del vector. Otra desventaja de esta representación es que permite poca flexibilidad, ya que cada posición del vector de horarios representa un día y una hora en particular, sin embargo, si se requiere cambiar la cantidad de bloques horarios de una sede, ya sea aumentando o disminuyéndolos, se debe realizar cambios mayores incluso al algoritmo.

Principalmente por el motivo de la flexibilidad es que se opta por un diseño de cromosoma como el que se describe a continuación en la Figura 25.

CROMOSOMA				
ID SECCIÓN	ID DOCENTE	ID SALA	DIA	HORA
1	7	23	1	1
1	7	23	1	2
1	7	23	1	3
1	7	23	1	4
2	45	12	2	7
2	45	12	2	8
3	6	45	5	12
3	6	45	5	13
3	6	45	5	14
4	3	5	4	5
4	3	5	4	6
4	3	5	1	1
..	..	..	..	..
n	6	2	3	2

Figura 25: Representación del cromosoma.

En esta representación se identifica el id de cada sección, el cual permanece fijo y el algoritmo asigna las tres variables de la planificación, id de docente, id de sala y horario. El horario está conformado por el código del día donde 1 es lunes, 2 es martes y así sucesivamente y el código del bloque horario. Para cada sección existen tantos registros como bloques horarios semanales sean necesarios.

Como el algoritmo genera tantos cromosomas como se defina según el tamaño de la población y la cantidad de generaciones, se hace necesario identificar cada cromosoma y almacenar datos relevantes que deben preservarse durante la ejecución del algoritmo. Se define la siguiente estructura como cabecera de cromosoma.

CABECERA DE CROMOSOMA					
ID CROMOSOMA	OBJETIVO	GENERACION	ID PADRE	ID MADRE	MEJOR SOLUCION
1	60%	1			0
2	35%	1			0
3	25%	1			0
..	..	..	..	..	..
n	100%	100	1	3	1

Figura 26: Representación de cabecera de cromosoma.

Los datos de la cabecera del cromosoma son:

ID Cromosoma: Identificador único del cromosoma dentro de la ejecución del algoritmo genético.

Objetivo: Representa el valor de la función objetivo para ese cromosoma.

Generación: Identificador de la generación a la cual pertenece un cromosoma. Útil para determinar las cruzas.

ID Padre: Identificador del cromosoma padre que dio origen al cromosoma.

ID Madre: Identificador del cromosoma madre que dio origen al cromosoma.

Mejor solución: Flag que indica que el cromosoma es o no la mejor solución.

4.6 Función objetivo

Se define la siguiente función objetivo:

$$fo = 100 * \frac{(n - n_c)}{n}$$

Dónde:

n = cantidad de secciones a asignar

n_c = cantidad de secciones que presentan algún tipo de choque horarios.

Para determinar n_c, se contabilizan las secciones que presentan algún incumplimiento de una restricción, como:

- Presenta planificación en una sala que se encuentra utilizada por otra sección en el mismo instante de tiempo.
- Presenta planificación con un docente que se encuentra utilizado por otra sección en el mismo instante de tiempo.
- Presenta planificación con un docente que se encuentra imposibilitado de realizar la clase en el horario definido para la sección.
- Presenta planificación que coincide con la planificación de otra sección y cuyos alumnos son comunes.

La función objetivo siempre presentará valores en el intervalo [0,100], donde 100 representa soluciones óptimas.

4.7 Algoritmo

La estructura principal del algoritmo genético desarrollado se presenta en la Figura 27.

```
PROCEDURE algoritmoGenetico AS

BEGIN

  preparaDatosEntrada();
  inicializaTablasTrabajo();

  primeraGeneracion();

  FOR i in 2 .. c_generaciones LOOP

      IF v_encontro_solucion = 0 THEN

         preservaMejores();

         FOR j in 1..(c_tamaño_poblacion)/2 LOOP

                seleccionaPadres();
                cruzaSoluciones();
                mutaSoluciones();
                calculaObjetivo();

         END LOOP;

         guardaMejorSolucion();

         IF v_objetivo = c_objetivo THEN
            v_encontro_solucion :=1;
         END IF;

         borraTablasTrabajo();
         liberaMemoria();

      END IF;
   END LOOP;
END;
```

Figura 27: Pseudocódigo del algoritmo genético.

A continuación se describen los elementos principales del algoritmo.

4.7.1 Preparación de datos de entrada

Corresponde a la extracción de datos desde la fuente, es decir el sistema académico SIGA. En esta etapa se completan las estructuras definidas en las secciones 4.3 y 4.4 basándose en los datos de la planificación académica

realizada por el Director de carrera, la disponibilidad ingresada por los docentes, la disponibilidad de salas y si existieran los datos de una planificación parcial ya realizada (ver Figura 28).

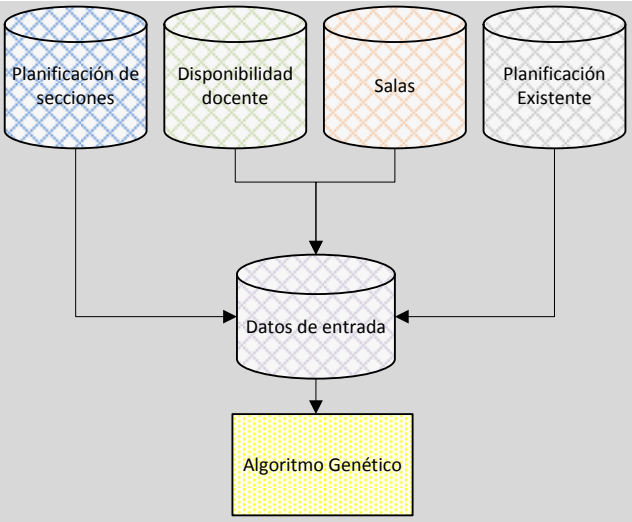


Figura 28: Datos de entrada

4.7.2 Primera generación o población inicial

La población inicial puede ser generada mediante alguna heurística o tomando como base alguna solución ya existente para el problema, sin embargo, lo más común es que la población inicial se genere de forma aleatoria.

En este caso se escoge una primera generación o población inicial de forma aleatoria con lo cual se corre el riesgo de que la población inicial sea de una baja calidad y atente contra la rápida convergencia del algoritmo. Una población inicial generada mediante heurística podría degradar el rendimiento del algoritmo consumiendo tiempo importante en buscar buenas soluciones iniciales.

La asignación de la población inicial considera que:

- A una sección se le debe asignar siempre el mismo docente en todos sus horarios.

- Los bloques horarios de una misma sección que son contiguos se asignan a la misma sala para que los alumnos no se cambien de sala entre clases.
- Se asignan solo docentes con disponibilidad horaria para la sección.
- Para una sección se asigna como máximo un conjunto de bloques horarios por días, evitando tener clases dos o más veces por día.

En el siguiente capítulo se muestran los ajustes a la selección de la población inicial en base a los resultados de la ejecución del algoritmo.

4.7.3 Preservación de mejores soluciones

Como a continuación se ve en uno de los puntos siguientes, se opta por una cruza destructiva, es decir se acepta como válidos los nuevos individuos a pesar de que su *fitness* o función objetivo no sea necesariamente mejor que la de sus progenitores. Sin embargo, para garantizar que los mejores individuos, según su función objetivo, se propaguen a las siguientes generaciones, se define que un porcentaje de los individuos, los mejores según su función objetivo, de una población se traspasen directamente a la siguiente generación pero sufriendo una mutación con una cierta probabilidad definida inicialmente como cero o con un valor muy bajo.

4.7.4 Selección de padres o progenitores

El tipo de selección escogido corresponde a un torneo determinístico ya que se escoge al azar dos individuos y se hace competir comparando su función objetivo. De los dos, el que tenga mejor función objetivo se escoge como "padre". Del mismo modo se escoge una "madre" y se procede a la cruza entre ellos. El algoritmo se asegura que una pareja de individuos solo se cruce una vez. El único individuo no apto para participar de una cruza es el peor individuo de su generación, según su función objetivo, ya que podría ser escogido como candidato pero perdería siempre el torneo. Este tipo de

cruza no asegura que los mejores individuos sean escogidos para dar origen a la nueva generación.

4.7.5 Función de cruza y mutación

El tipo de cruza escogido corresponde a una cruza de un punto y de tipo destructiva. Es decir, la combinación de los dos padres escogidos se realiza en un punto definido al azar y a partir de ellos se obtienen dos descendientes que se incorporan a la población a pesar de que su función objetivo no sea necesariamente mejor que la de sus padres.

Se grafica la función de cruza en la Figura 29.

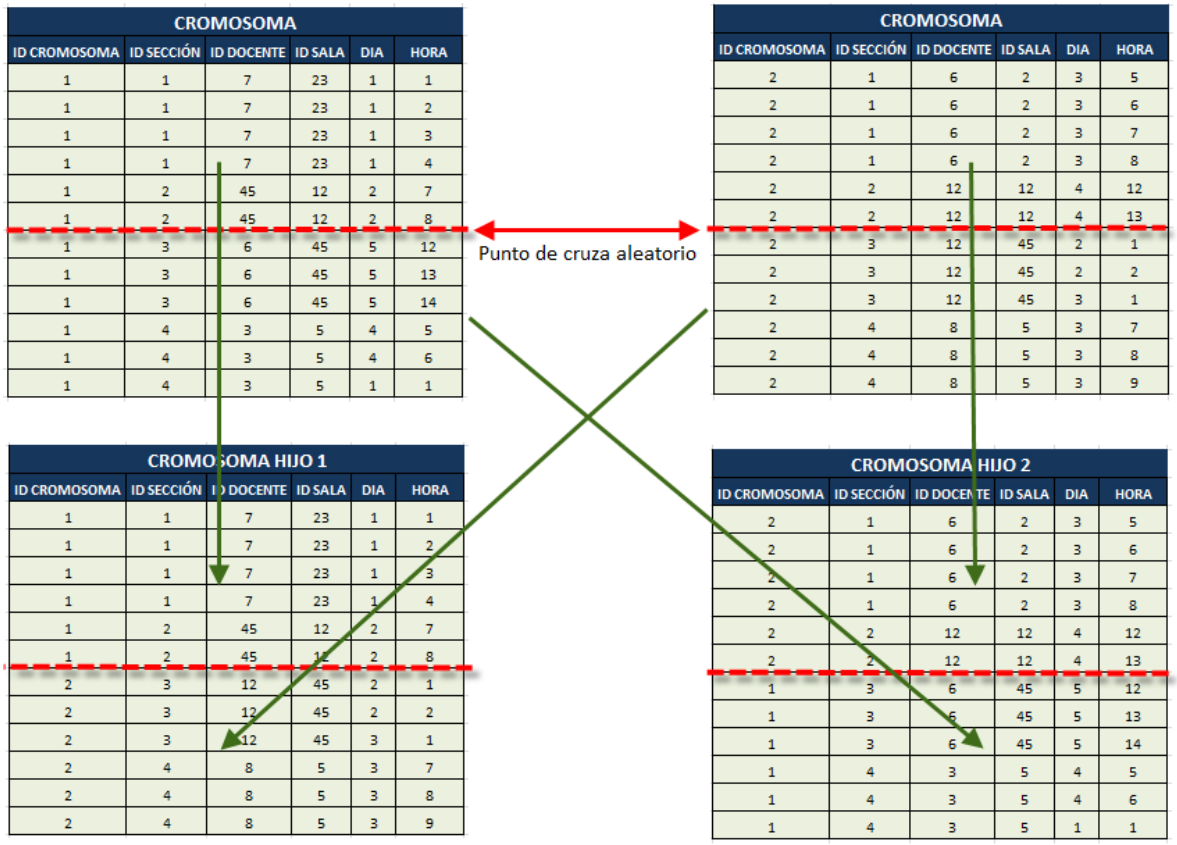
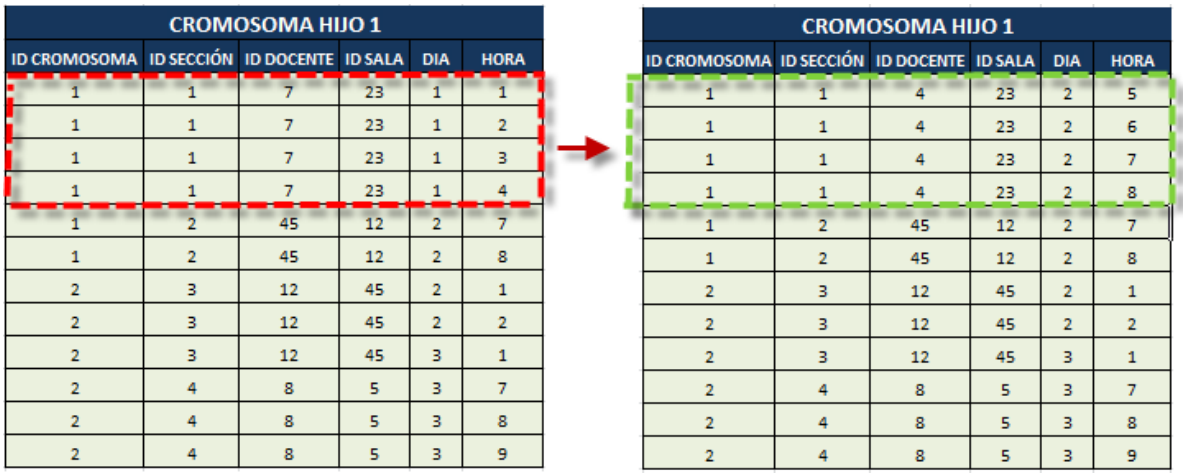


Figura 29: Función de cruza.

El punto de cruza si bien es escogido aleatoriamente, cuida que no se divida la planificación de una sección, ya que esto podría generar horarios con más o menos horas de lo que se requiere para una sección.

La operación de mutación consiste en calcular aleatoriamente un factor de mutación de 1 a 100, si ese factor es menor o igual al mínimo definido como probabilidad de mutación, se aplica un cambio al gen correspondiente a esa sección modificando la sala, el docente y el horario aleatoriamente, pudiendo quedar con los mismos parámetros que tenía inicialmente. Esta operación se repite por cada uno de los genes de un cromosoma, tanto para los cromosomas provenientes de una cruza y aquellos que son preservados de la generación anterior.



CROMOSOMA HIJO 1					
ID CROMOSOMA	ID SECCIÓN	ID DOCENTE	ID SALA	DIA	HORA
1	1	7	23	1	1
1	1	7	23	1	2
1	1	7	23	1	3
1	1	7	23	1	4
1	2	45	12	2	7
1	2	45	12	2	8
2	3	12	45	2	1
2	3	12	45	2	2
2	3	12	45	3	1
2	4	8	5	3	7
2	4	8	5	3	8
2	4	8	5	3	9

CROMOSOMA HIJO 1					
ID CROMOSOMA	ID SECCIÓN	ID DOCENTE	ID SALA	DIA	HORA
1	1	4	23	2	5
1	1	4	23	2	6
1	1	4	23	2	7
1	1	4	23	2	8
1	2	45	12	2	7
1	2	45	12	2	8
2	3	12	45	2	1
2	3	12	45	2	2
2	3	12	45	3	1
2	4	8	5	3	7
2	4	8	5	3	8
2	4	8	5	3	9

Figura 30: Función de mutación.

4.7.6 Parámetros del algoritmo genético

Existen algunos parámetros importantes para el algoritmo genético que deben permitir su ajuste de acuerdo a los resultados que se obtengan en la ejecución del algoritmo.

Se define como un parámetro el tamaño de la población inicial, el cual se mantiene constante en las siguientes generaciones. El valor del tamaño de la población inicial se establece primeramente en un valor de 8, es decir se crearán ocho cromosomas en cada generación. Pero el algoritmo es flexible en esa cantidad y permite su ajuste según las necesidades.

La probabilidad de mutación se establece inicialmente en un 2%, es decir un cromosoma se muta solo en un 2% de las veces, cambiando aleatoriamente

una sección de sala, docente y horario. El parámetro de probabilidad se puede ajustar de acuerdo a los resultados de la ejecución del algoritmo.

También se define como un parámetro del algoritmo genético la cantidad de generaciones para las cuales itera, siendo ésta además una condición de término del algoritmo, es decir, finaliza una vez que se complete la cantidad de generaciones independiente de la función objetivo obtenida. Se establece inicialmente 200 generaciones como máximo de iteración.

La segunda condición de término queda establecida y parametrizada por un valor esperado de la función objetivo. Inicialmente se espera que la función objetivo alcance un 100%, es decir, que no existan choques horarios entre salas, docentes y cursos. Para ejecuciones del algoritmo con muchos cursos o con condiciones muy "ajustadas" como pocas salas o pocos docentes, esta condición de término puede bajarse a un 90% o incluso menos.

5 RESULTADOS

5.1 Caso de estudio

Para efectos experimentales se considera como población de prueba los datos reales de la sede Coyhaique de INACAP, la sede con menor cantidad de alumnos, docentes y salas en la institución. Se utilizan los datos de una muestra de la planificación académica del primer semestre del año 2014. La muestra está compuesta por 100 secciones de ambas jornadas, 50 de jornada diurna y 50 de jornada vespertina, 84 profesores y 40 salas. Para cada profesor ya se tiene definida su disponibilidad en cuanto a horario y en cuanto a las asignaturas que puede impartir. Del mismo modo, está definida la disponibilidad de las salas. Las secciones se encuentran organizadas en 46 cursos que tiene entre 1 y 6 secciones las cuales no deben generar choques de horario entre ellas. El diagrama del caso de estudio se muestra en la Figura 31.

El horario está conformado por 12 bloques horarios diarios en jornada diurna, que da origen a 60 bloques semanales en jornada diurna y 6 bloques horarios diarios en jornada vespertina que dan origen a 30 bloques semanales en jornada vespertina.

Cada sección tiene a lo menos dos docentes que puede impartir la clase, teniendo incluso secciones con hasta diez docentes disponibles. A su vez cada sección tiene al menos 13 salas disponibles según su capacidad y algunas secciones pueden dictarse en cualquiera de las 40 salas existentes. Las secciones escogidas tienen una frecuencia de clases semanales que va desde uno a tres días y desde dos a seis horas semanales.

Los docentes tienes restricción de asignación de 60 bloques horarios semanales y 12 bloques horarios diarios. La disponibilidad horaria de los docentes va desde docentes que tienen solo 5 horas semanales en una jornada, hasta docentes con 72 horas semanales en jornada diurna y 36 horas en jornada vespertina, lo cual representa una disponibilidad completa.

Las secciones se encuentran organizadas en cursos los cuales tienen como conjunto un máximo de 27 horas a asignar en jornada diurna, de un máximo de 60 bloques diurnos y 16 bloques en jornada vespertina, de un máximo de 30 bloques vespertinos.

Se asume que no existe una planificación parcial realizada, es decir, se cuenta con todas las salas disponibles y todos los docentes disponibles en todos los horarios indicados por ellos.

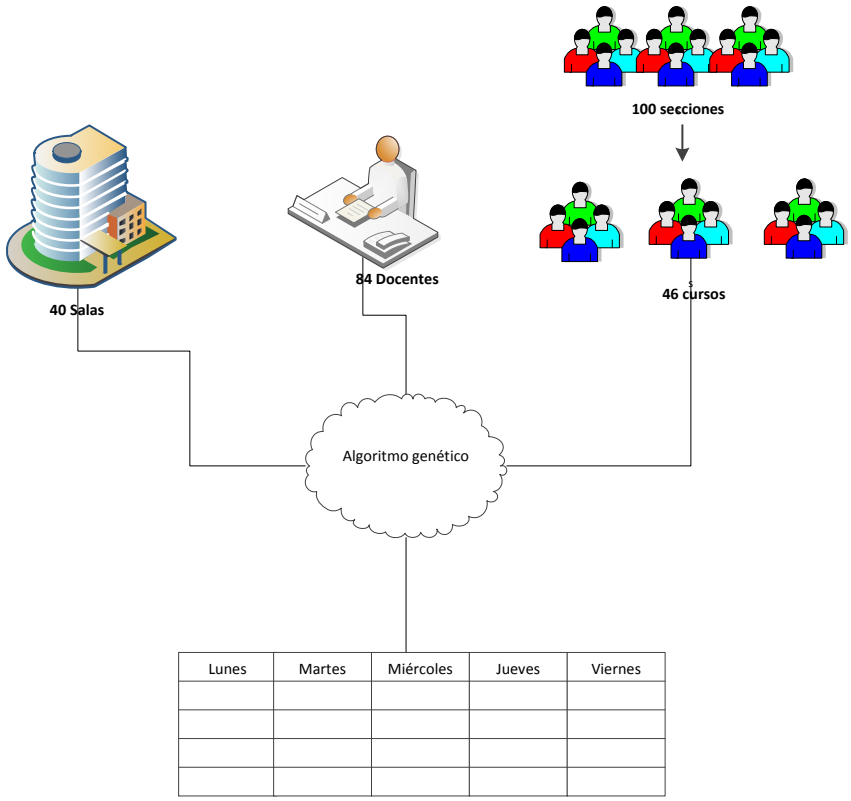
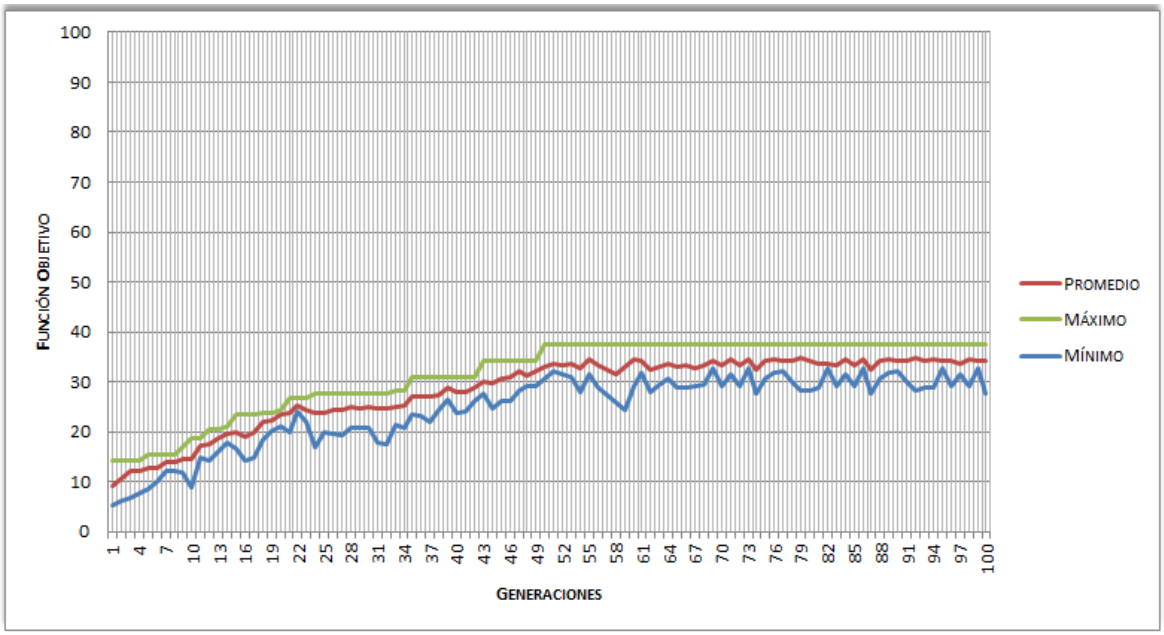


Figura 31: Diagrama del caso de estudio.

5.2 Ejecuciones preliminares del algoritmo

Con los datos descritos en la sección 5.1 y considerando los parámetros del diseño del algoritmo indicados en el capítulo anterior, se realizan ejecuciones del algoritmo, que permiten obtener algunas conclusiones que llevan a realizar ajustes en el algoritmo genético.

A continuación se muestra un gráfico (ver Figura 32) que representa el comportamiento del algoritmo con los parámetros iniciales. Se realizaron múltiples ejecuciones pero este gráfico corresponde a una que representa el comportamiento de la mayoría de las ejecuciones.



Parámetros

Cantidad de Secciones	100
Cantidad de Generaciones	200
Probabilidad de mutación	2%
Condición de término	100%

Figura 32: Gráfico de ejecuciones preliminares del algoritmo genético.

En el gráfico anterior se muestra en el eje Y el valor alcanzado por la función objetivo en la ejecución del algoritmo y en el eje X las generaciones. Se grafica en color rojo el promedio de la función objetivo obtenido de las 8 soluciones que componen cada generación, en color verde el máximo valor de la función objetivo alcanzado y en color azul el mínimo valor de la función objetivo. Se presenta el gráfico solo hasta la generación 100, ya que el comportamiento entre la generación 100 y la 200 es prácticamente igual, solo con leves variaciones.

Se observan los siguientes problemas:

Mala calidad de la población inicial: Al no utilizar ninguna heurística o estrategia de asignación de la población inicial las soluciones de la primera generación son de mala calidad, alcanzando solo un 10% de promedio lo cual significa que un 90% de las secciones presenta choques horarios u horarios no válidos. Esta condición de inicio perjudica el desempeño del algoritmo, debiendo repuntar mucho para alcanzar el objetivo.

No convergencia: A partir de la generación 50 el algoritmo logra su máximo, siendo por debajo del 40%, es decir, una solución de baja calidad. Luego se mantiene en ese máximo sin lograr mejorar en las siguientes iteraciones. Al analizar los cromosomas se visualiza que las soluciones encontradas son muy parecidas entre sí, teniendo poca variedad en la población, con lo cual la cruza no tiene efectos ya que mezcla cromosomas muy similares o iguales dejando la mejora solo dependiendo de la operación de mutación. Pero al tener una probabilidad baja de mutación no se logra la mejora esperada. Otro motivo por el cual se obtiene poca diversidad es por la preservación de los mejores cromosomas entre una generación y otra, sin embargo, esto asegura que el algoritmo siempre muestre una tendencia al alza ya que su máximo se mantendrá, por lo menos, en la generación siguiente.

Alto tiempo de ejecución: El tiempo de ejecución del algoritmo en promedio desde su inicio hasta término sin éxito (debido a que se iteró hasta las 200 generaciones definidas) sobrepasó los 60 minutos y según lo analizado se fue degradando a medida que avanzaba, debido a que debía manejar una gran cantidad de datos. Si bien el tiempo empleado es relativamente bueno para este tipo de problemas, se considera alto ya que la escala de la institución obliga a tratar de asignar sedes con más de mil secciones mientras que la muestra es solo de cien secciones.

Alto uso de espacio de almacenamiento: En las ejecuciones piloto se detectó un alto uso de espacio de almacenamiento debido a la gran cantidad

de registros que se van generando a lo largo de la ejecución del algoritmo, lo cual a medida que avanza el algoritmo va haciendo más difícil la selección y los cálculos de la función objetivo. Con la muestra de cien secciones no es tan evidente el uso de almacenamiento pero al extrapolar a una situación real con mil secciones o más el almacenamiento es un punto que se debe considerar.

5.3 Ajustes al algoritmo genético

Para mejorar los problemas presentados en las ejecuciones preliminares del algoritmo genético se realiza una serie de ajustes de diseño y de los parámetros definidos.

La calidad de la población inicial influye directamente en el tiempo empleado para llegar a una solución óptima y considerando que con una muestra de 100 secciones se obtiene una primera generación con un promedio de 10% en su función objetivo, se espera que con una población mayor ese promedio sea aún más bajo debido a que podría aumentar la choques horarios. Para aumentar la calidad de la población inicial se incorpora un proceso de iteración al seleccionar los elementos para una sección, es decir, se ordenan las secciones según su disponibilidad de docentes y/o salas, de menor a mayor, asignando en primer lugar las secciones con menos posibilidades de elección entre docentes y salas. Cada vez que se selecciona una sala, un docente y un horario para una sección, se chequea que no genere un choque horario con alguna sección ya asignada previamente; en caso de provocar un choque horario ya sea de docente o de sala, el algoritmo intenta con otra de las opciones de docente y/o sala tratando de evitar el choque horario. Este proceso iterativo se asume costoso y podría degradar aún más el rendimiento del algoritmo. Se suma como un parámetro del algoritmo la cantidad de veces que itera al generar la población inicial pero se revisa de forma empírica hasta que valor es conveniente subir ese parámetro sin provocar un alza descontrolada del tiempo de ejecución del algoritmo. Se

concluye que el algoritmo itera entre 5 y 10 veces para cada asignación de sala y/o docente, ya que al subir ese parámetro a un valor mayor a 10 no se obtiene una mejora sustancial en la calidad de la función objetivo en la población inicial y si se degrada el tiempo empleado en generar la población inicial.

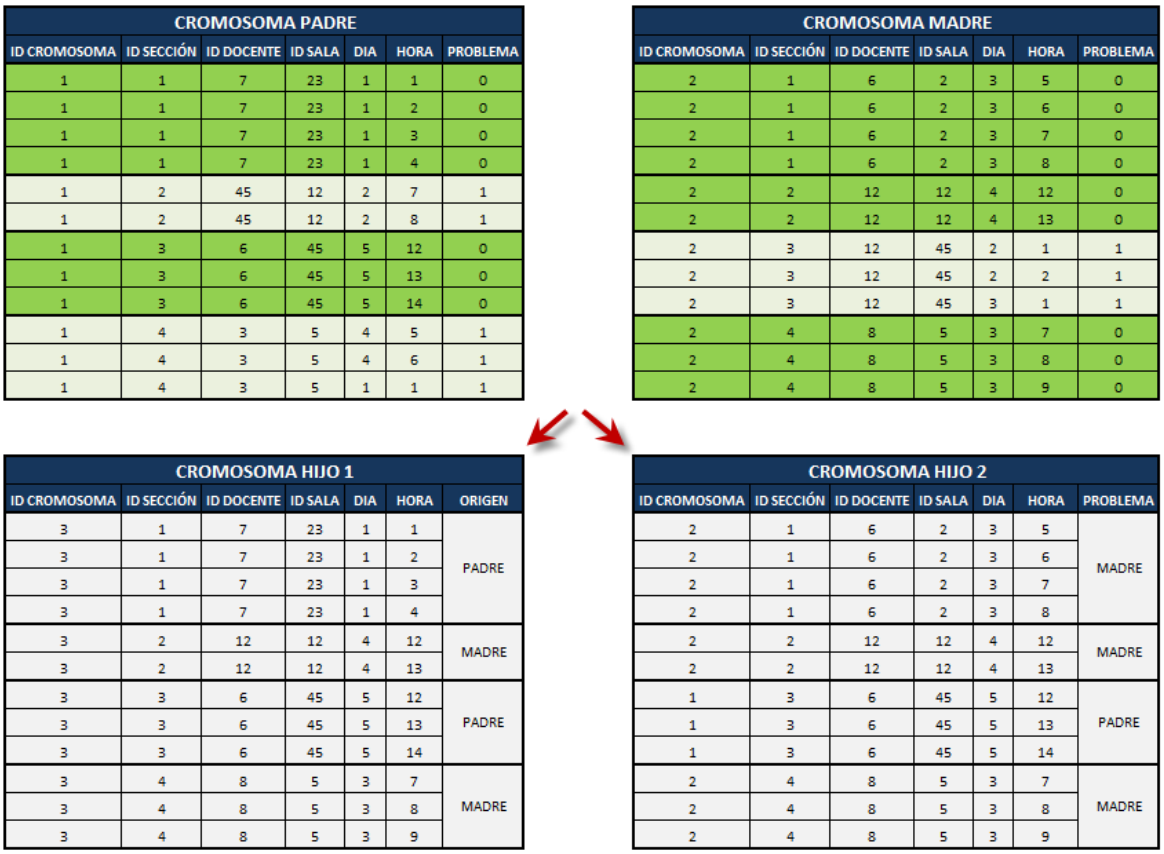
El tamaño de la población establecido inicialmente en ocho cromosomas provoca que a medida que el algoritmo itera las soluciones sean muy similares entre si. Por esto se modifica el tamaño de la población, iterando entre 10 y 100 soluciones y se determina que un número apropiado para esta instancia del problema es entre 30 y 50 cromosomas por generación, lo cual asegura diversidad, no impacta altamente en el tiempo de iteración del algoritmo y permite llegar a varias mejores soluciones diferentes o en su defecto varias soluciones cercanas a la mejor solución que podrían ser soluciones satisfactorias. Se mantiene la preservación de las dos mejores soluciones de cada generación pero modificando la mutación aplicada a estos cromosomas, mutando con una cierta probabilidad, pero solo aquellas secciones que presentan problemas de choques horarios ya sea de docente y/o sala. Para esto se identifica en cada cromosoma si la sección corresponde a una sección sin problemas dentro de la solución o si presenta algún tipo de planificación no viable. Luego para cada gen con problemas se aplica una mutación con una probabilidad del 2%, tras lo cual el cromosoma podría permanecer igual, podría deshacer el problema de choque horario o podría provocar otro choque horario con otra sección.

CROMOSOMA						
ID CROMOSOMA	ID SECCIÓN	ID DOCENTE	ID SALA	DIA	HORA	PROBLEMA
2	1	6	2	3	5	0
2	1	6	2	3	6	0
2	1	6	2	3	7	0
2	1	6	2	3	8	0
2	2	12	12	4	12	1
2	2	12	12	4	13	1
1	3	6	45	5	12	0
1	3	6	45	5	13	0
1	3	6	45	5	14	0
1	4	3	5	4	5	1
1	4	3	5	4	6	1
1	4	3	5	1	1	1

Figura 33: Cromosoma preservado.

En la Figura 33 se observa un cromosoma preservado en el cual se destaca aquellas secciones que presentan problemas de planificación, ya sea entre ellas o con cualquier otra del cromosoma. Las secciones que presentan problemas son candidatos a mutación. Si la probabilidad obtenida al azar es menor o igual a la probabilidad definida de mutación se procede a mutar. La probabilidad de mutación durante la preservación se maneja como un parámetro distinto de la mutación de cromosomas generados a través de la función de cruza.

Aprovechando que se identifica las secciones que presentan choques horarios, se experimenta con un tipo de cruza un poco más dirigida y no completamente al azar, ya que la cruza al azar provoca que la convergencia sea lenta. Esta nueva cruza dirigida se realiza preservando en el cromosoma todas las secciones que no presentan problemas y se obtiene del otro progenitor la planificación de las secciones que presentan problemas, independientemente si en el otro progenitor presentan o no presentan problemas.



CROMOSOMA PADRE

ID CROMOSOMA	ID SECCIÓN	ID DOCENTE	ID SALA	DIA	HORA	PROBLEMA
1	1	7	23	1	1	0
1	1	7	23	1	2	0
1	1	7	23	1	3	0
1	1	7	23	1	4	0
1	2	45	12	2	7	1
1	2	45	12	2	8	1
1	3	6	45	5	12	0
1	3	6	45	5	13	0
1	3	6	45	5	14	0
1	4	3	5	4	5	1
1	4	3	5	4	6	1
1	4	3	5	1	1	1

CROMOSOMA MADRE

ID CROMOSOMA	ID SECCIÓN	ID DOCENTE	ID SALA	DIA	HORA	PROBLEMA
2	1	6	2	3	5	0
2	1	6	2	3	6	0
2	1	6	2	3	7	0
2	1	6	2	3	8	0
2	2	12	12	4	12	0
2	2	12	12	4	13	0
2	3	12	45	2	1	1
2	3	12	45	2	2	1
2	3	12	45	3	1	1
2	4	8	5	3	7	0
2	4	8	5	3	8	0
2	4	8	5	3	9	0

CROMOSOMA HIJO 1

ID CROMOSOMA	ID SECCIÓN	ID DOCENTE	ID SALA	DIA	HORA	ORIGEN
3	1	7	23	1	1	PADRE
3	1	7	23	1	2	
3	1	7	23	1	3	
3	1	7	23	1	4	
3	2	12	12	4	12	MADRE
3	2	12	12	4	13	
3	3	6	45	5	12	PADRE
3	3	6	45	5	13	
3	3	6	45	5	14	
3	4	8	5	3	7	MADRE
3	4	8	5	3	8	
3	4	8	5	3	9	

CROMOSOMA HIJO 2

ID CROMOSOMA	ID SECCIÓN	ID DOCENTE	ID SALA	DIA	HORA	PROBLEMA
2	1	6	2	3	5	MADRE
2	1	6	2	3	6	
2	1	6	2	3	7	
2	1	6	2	3	8	
2	2	12	12	4	12	MADRE
2	2	12	12	4	13	
1	3	6	45	5	12	PADRE
1	3	6	45	5	13	
1	3	6	45	5	14	
2	4	8	5	3	7	MADRE
2	4	8	5	3	8	
2	4	8	5	3	9	

Figura 34: Cruza dirigida.

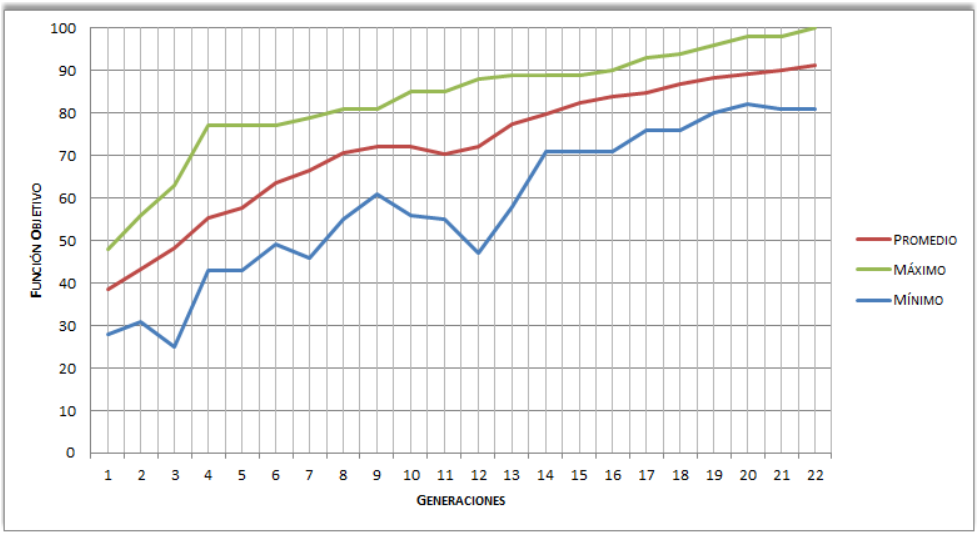
En la Figura 34 se grafica una cruza dirigida. En los cromosomas padre y madre se destacan en color verde aquellas secciones que no presentan ningún problema de planificación. En el cromosoma denominado Hijo 1, se conservan todas las secciones correctas desde el padre y se obtiene las secciones con problemas desde el cromosoma madre, y análogamente en el cromosoma Hijo 2 se conservan todas las secciones que no presentan problemas en el cromosoma madre y se obtienen desde el cromosoma padre las secciones que presentan problemas. Se experimenta ejecutando el algoritmo exclusivamente con cruza dirigida, con lo cual se aprecia que el algoritmo tiende a converger rápidamente. Sin embargo, en algunas ejecuciones el algoritmo efectivamente converge rápidamente, pero al estar cerca de una solución óptima presenta pocas mejoras. Se opta por no eliminar la cruza aleatoria ya que entrega mayor diversidad a las soluciones

dejando el algoritmo con la opción de ejecutar ambos tipos de cruza definiendo una probabilidad para cada una de ellas. Inicialmente se define una probabilidad de 50% para cada una.

Para evitar problemas con el almacenamiento y para mejorar el tiempo de ejecución del algoritmo se establece una destrucción de los cromosomas de generaciones ya usadas, dejando solo datos estadísticos pero no su detalle del horario generado. Esto obliga a ir almacenando en una estructura adicional las mejores soluciones para evitar su borrado. Cada vez que el algoritmo detecta que encontró una solución igual o mejor a la de mejor fitness disponible actualmente se guarda esa mejor solución liberando con eso el espacio con lo cual se mejora el tiempo de respuesta en los cálculos.

5.4 Ejecución de algoritmo

Con los ajustes realizados al algoritmo genético se logra que el algoritmo cumpla su objetivo, es decir, que entregue una solución óptima para la muestra de datos seleccionada. A continuación la Figura 35 muestra un gráfico que resume las ejecuciones del algoritmo ajustado.



Parámetros	
Cantidad de Secciones	100
Cantidad de Generaciones	200
Probabilidad de mutación	2%
Condición de término	100%

Figura 35: Gráfico de ejecuciones finales del algoritmo genético.

Como se puede ver en el gráfico la población inicial parte con un promedio cercano a un 40% en su función objetivo, siendo incluso las peores soluciones (que están levemente bajo el 30%) mejores al máximo alcanzado en la primera generación obtenido con la versión anterior del algoritmo, es decir, sin una iteración simple para la primera generación. Al realizar una iteración simple para crear la población inicial y no una heurística compleja, el impacto en el tiempo de ejecución del algoritmo es bajo y tiene un alto impacto en la cantidad de iteraciones o generaciones que se deben realizar para que el algoritmo converja.

Al aumentar el tamaño de la población se evitan los problemas de baja convergencia del algoritmo genético, ya que la diversidad de soluciones implica que el algoritmo no converja hacia soluciones iguales. Se aprecia en el gráfico de la Figura 35 que ya en la generación 15 el algoritmo encuentra soluciones con 90% de secciones sin problemas de choques horarios y en poco más de 20 generaciones encuentra una solución óptima.

Además, la calidad general de las soluciones mejora con los ajustes. A continuación se muestra un gráfico de la función objetivo de las soluciones de la última generación. En él podemos observar que todas las soluciones se encuentran sobre el 80% según su función objetivo y más de la mitad de las soluciones sobre el 90% de su función objetivo. Incluso se entregan cuatro soluciones con el 97% de secciones con planificación correcta, las cuales pueden ser alternativas a considerar como solución final.

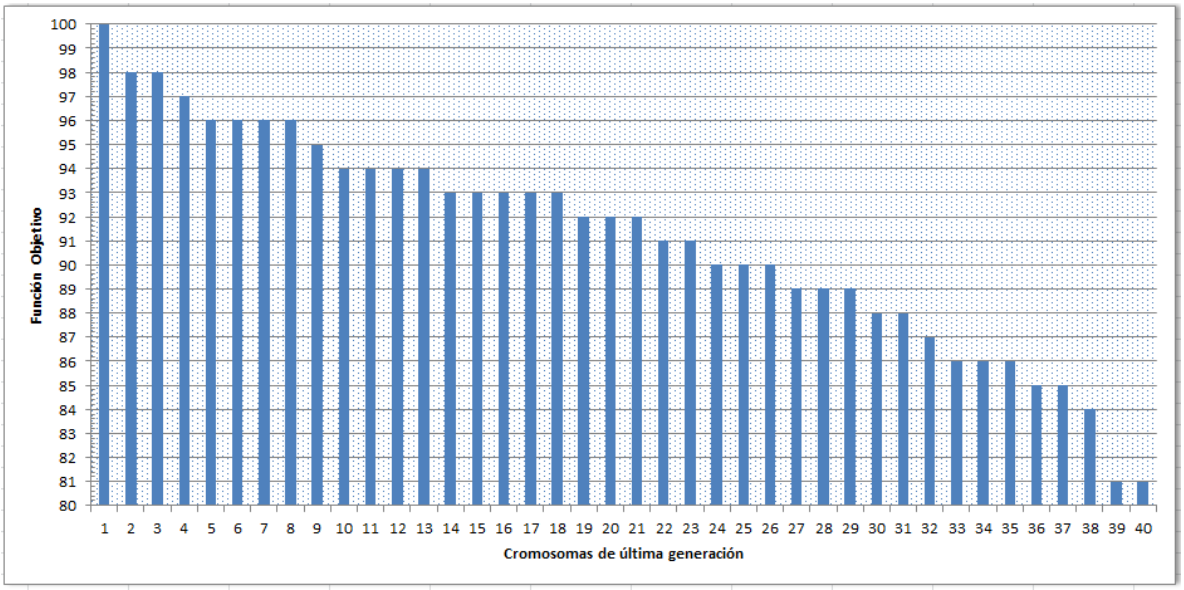


Figura 36: Gráfico de función objetivo de última generación.

En cuanto al tiempo de ejecución del algoritmo, este se ve disminuido por los ajustes realizados, alcanzando una solución óptima en aproximadamente cuatro minutos.

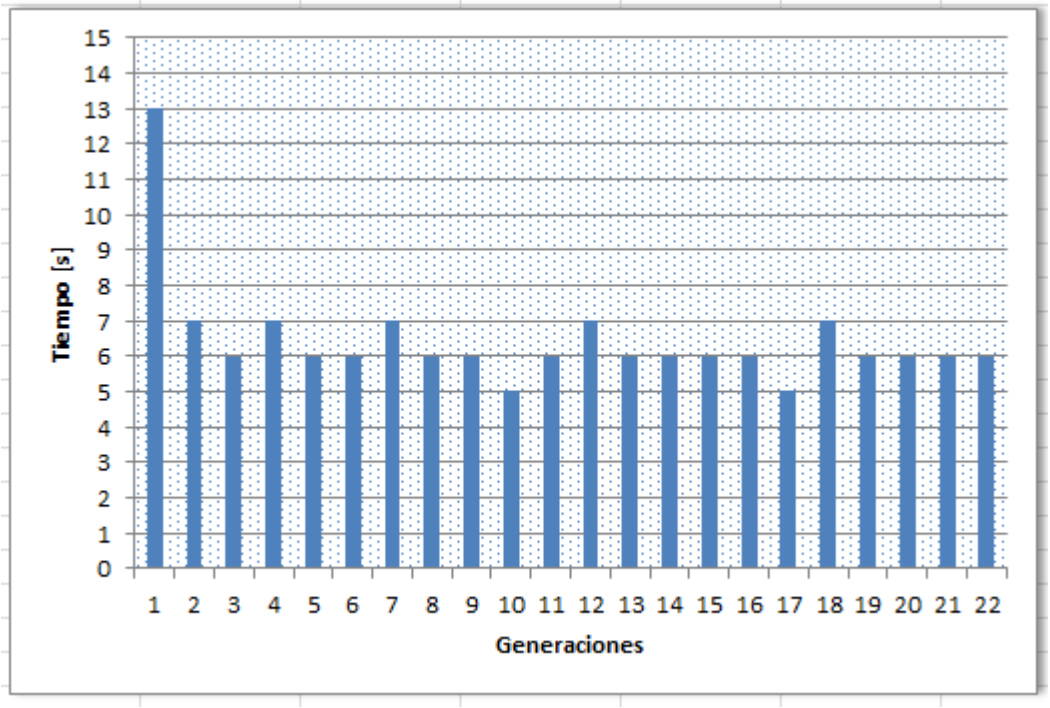


Figura 37: Gráfico de tiempo de ejecución por cada generación.

En el gráfico de la Figura 37 se muestra que la primera generación demora aproximadamente 13 segundos en generarse debido a la iteración inicial mientras que las siguientes generaciones en promedio se construyen en solo seis segundos.

Al destruir las soluciones que no son utilizadas en el futuro se mejora tanto el tiempo de ejecución como el alto uso de almacenamiento.

5.5 Visualización gráfico del horario generado

Para visualizar el horario generado, se integra al sistema académico SIGA y se puede apreciar gráficamente la calidad del horario entregado por el algoritmo genético. Esto se aprecia en la Figura 38.

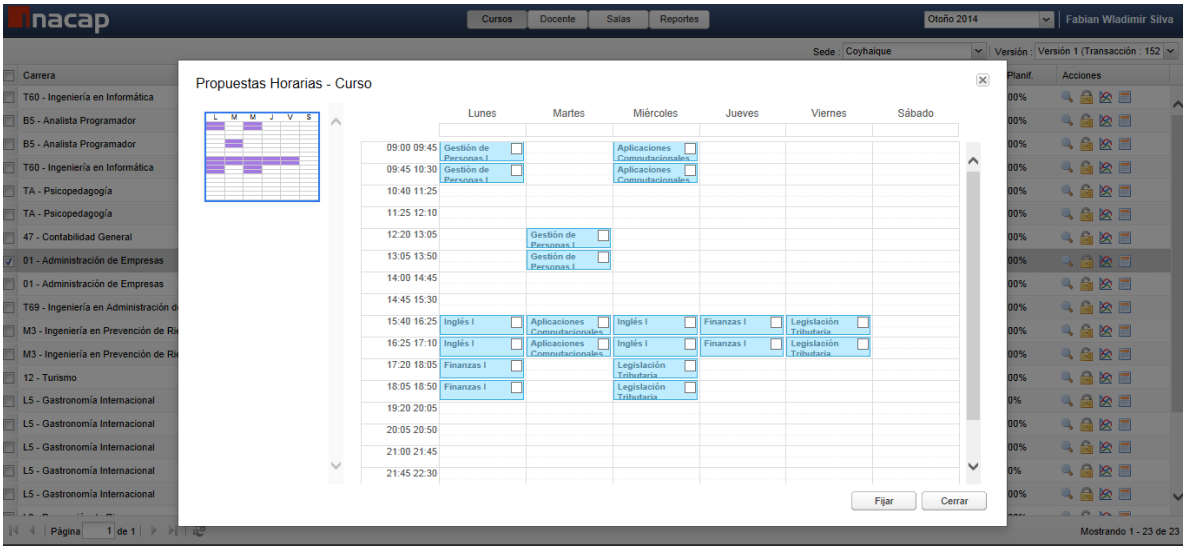


Figura 38: Ejemplo gráfico de horario entregado por el algoritmo genético.

6. CONCLUSIÓN

El principal objetivo de esta investigación que consiste en construir una solución que ayude a reducir el tiempo destinado a la creación de horarios académicos en INACAP se cumplió en efecto gracias a este trabajo dado que ahora se cuenta con una herramienta que realiza un proceso, que antes tardaba semanas o días en el mejor de los casos, en minutos u horas, lo cual representa un sustancial ahorro de recursos. Además, producto de la rapidez con que se obtiene un horario permite iterar sucesivas veces hasta encontrar un mejor horario para los alumnos.

La herramienta asegura que los horarios generados cumplen con las restricciones impuestas y entrega la flexibilidad de poder realizar asignaciones horarias parciales.

Otro punto importante es que la solución entregada está completamente integrada con el SIGA (Sistema Integrado de Gestión Académica), ya que se alimenta de los datos de entrada de este y a su vez el horario se migra fácilmente hacia esta plataforma.

Además, la investigación permitió demostrar que para este tipo de problemas la aplicación de algoritmos genéticos es una práctica muy conveniente ya que entrega varias ventajas. La técnica de algoritmos genéticos se encuentra ampliamente documentada por lo cual el aprendizaje de esta no es un problema. Al ser una técnica estándar se puede poner en práctica rápidamente, aunque su modelamiento y los ajustes son fundamentales para alcanzar una solución al problema a resolver. Se pudo comprobar que los algoritmos genéticos son una técnica ideal para problemas de timetabling ya que producen soluciones rápidas y eficientes. Los resultados entregados por este tipo de algoritmos son muy aceptables.

La técnica de algoritmos genéticos entrega también mucha flexibilidad, ya que permite que se adapte a por ejemplo distintos horarios, distintos universos de salas, etc. Además, tienen la ventaja de entregar varias

soluciones. En este caso entrega una solución óptima y varias soluciones muy cercanas a la óptima.

Al comparar los resultados entregados por el algoritmo genético con soluciones obtenidas de forma manual se aprecia que ambos cumplen con las restricciones más duras del problema, la diferencia es el esfuerzo invertido en ello, ya que una solución manual tardó semanas mientras que la obtenida por esta herramienta tarda horas.

La solución tiene como punto más débil el cumplimiento de las restricciones suaves ya que si bien los puntos de asignación por prioridad son abordados en la primera generación de soluciones, en la iteración del algoritmo esa asignación prioritaria podría perderse por efecto de las mutaciones. También la minimización de "ventanas" es un proceso costoso de medir en una función objetivo y podría degradar el tiempo necesario para obtener la solución, por lo cual no se incluye en el algoritmo y se mide cuando la solución ya se encuentra generada.

Esta investigación ha generado el *know how* suficiente para hacer esta solución mucho más eficiente en el futuro y también para adaptarla a las necesidades que puedan surgir más adelante.

7. ANEXOS

7.1 Modelo de datos del algoritmo genético

El algoritmo genético se implementa en tablas relacionales y se programa en lenguaje PL/SQL. Si bien no es un lenguaje usual para este tipo de implementaciones, los buenos resultados obtenidos no hicieron necesario el cambio a otro lenguaje más apropiado.

A continuación, la Figura 39 muestra el modelo de tablas relacionales utilizado.

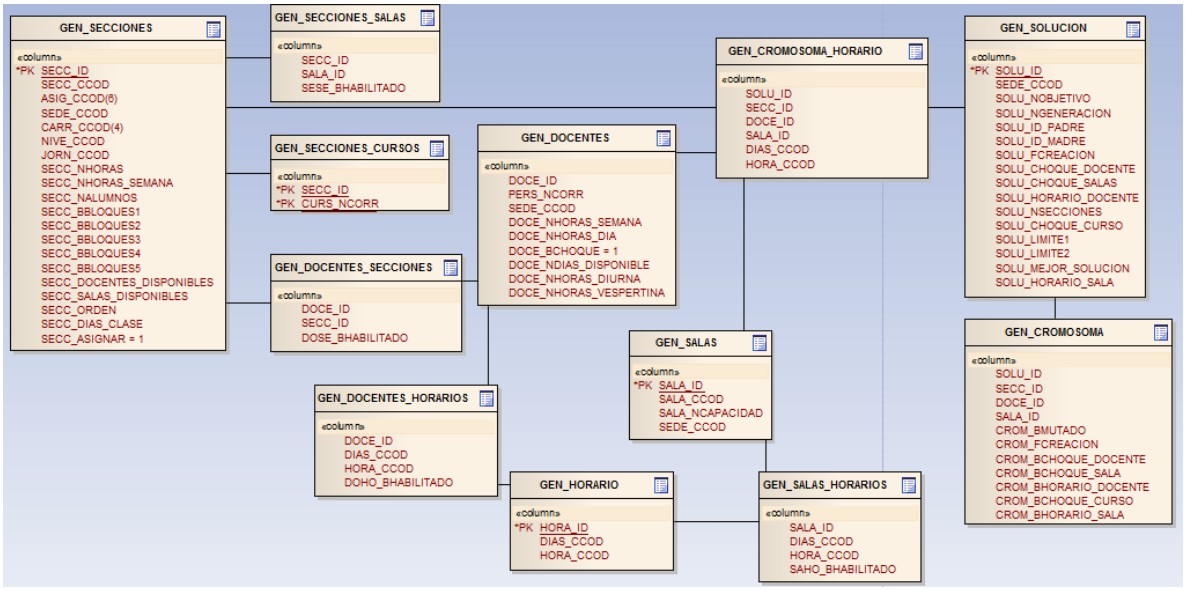


Figura 39: Modelo relacional.

7.2 Métodos del algoritmo genético

Los métodos implementados para el algoritmo genético son los siguientes:

Nombre	**Propósito**
algoritmoPrincipal	Estructura principal del algoritmo
borraTablasTrabajo	Eliminación de espacio

calculaObjetivo	Cálculo de fitness
cruzaAleatoria	Realiza operación de cruza de forma aleatoria en uno o dos puntos.
cruzaSolucion	Realiza operación de cruza dirigida.
generaCromosoma	Genera un cromosoma o solución
getAleatorio	Genera un número aleatorio
getAleatorioImpar	Genera un número aleatorio impar
getAleatorioPar	Genera un número aleatorio par
getDiaDocente	Obtiene un día en que un docente tenga disponibilidad horaria
getDocente	Obtiene un docente apto para una sección
getHorario	Obtiene un horario aleatorio para una sección
getHorarioDocente	Obtiene un bloque horario para un docente
getSala	Obtiene un sala que apta para una sección
guardaMejorSolucion	Guarda la mejor solución antes de realizar borrado de soluciones inválidas.
inicializaTablasTrabajo	Inicializa tablas de iteración.
insertaBloque	Inserta un bloque horario en la planificación de una sección
insertaCromosoma	Inserta un cromosoma (función de cruza)

insertaCromosomaLibre	Inserta un cromosoma (función de cruza)
liberaMemoria	Realiza liberación de espacio
limpiaBloque	Borra un bloque horario (mutación).
marcaObjetivo	Apoya el cálculo de la función objetivo
mutaSolucion	Mutación
mutaSolucionIncorrectos	Mutación dirigida a cromosomas preservados
obtieneCromosoma	Obtiene un cromosoma para cruza
preservaMejores	Preserva los mejores cromosomas de una generación a otra.
validaChoqueCurso	Valida si existe choque horario entre cursos.
validaChoqueDocente	Valida si existe choque horario entre docentes.
validaChoqueSala	Valida si existe choque horario entre salas.
validaHorarioDocente	Valida si existe asignación de horarios para docentes en horarios no permitidos.
validaHorarioSala	Valida si existe asignación de horarios para salas en horarios no permitidos.

8. BIBLIOGRAFÍA

[Baker 1974] Baker, K. R., & Baker, K. R. (1974). Introduction to sequencing and scheduling (Vol. 31). New York: Wiley.

[Beligiannis 2008] Beligiannis, G. N., Moschopoulos, C. N., Kaperonis, G. P., & Likothanassis, S. D. (2008). Applying evolutionary computation to the school timetabling problem: The Greek case. Computers & Operations Research, 35(4), 1265-1280.

[Bull 1993] Beasley, D., Martin, R. R., & Bull, D. R. (1993). An overview of genetic algorithms: Part 1. Fundamentals. University computing, 15, 58-58.

[Cook 1971] Cook, S. A. (1971, May). The complexity of theorem-proving procedures. In Proceedings of the third annual ACM symposium on Theory of computing (pp. 151-158). ACM.

[Cormen 2001] Leiserson, C. E., Rivest, R. L., & Stein, C. (2001). Introduction to algorithms. T. H. Cormen (Ed.). The MIT press.

[Eiben 2003] Eiben, A. E., & Smith, J. E. Introduction to evolutionary computing. 2003. ISBN 3540401849.

[Garey 1983] Garey, M. R., & Johnson, D. S. (1983). Crossing number is NP-complete. SIAM Journal on Algebraic Discrete Methods, 4(3), 312-316.

[Glover 1989] Glover, F. (1989). Tabu search—part I. ORSA Journal on computing, 1(3), 190-206.

[Goldberg 1989] Goldberg, D. E., & Holland, J. H. (1988). Genetic algorithms and machine learning. Machine learning, 3(2), 95-99.

[Gotlieb 1964] Csima, J., & Gotlieb, C. C. (1964). Tests on a computer method for constructing school timetables. Communications of the ACM, 7(3), 160-163.

[Granada 2006] Mauricio Granada E., Eliana M. Toro Ocampo, John F. Franco Baquero, "Programación óptima de horario de clase usando un algoritmo memético", Scientia Et Technica, vol. XII, núm. 30, mayo, 2006, pp. 255-260. Recuperado de: http://www.redalyc.org/articulo.oa?id=84920491051

[Hernández 2008] Rodrigo Hernández, Jaime Miranda P., Pablo A. Rey, "Programación de horarios de clases y asignación de salas para la Facultad de Ingeniería de la Universidad Diego Portales mediante un enfoque de programación entera", Revista Ingeniería de Sistemas Volumen XXII, Año 2008, pp. 121-141.

[Hilera 1995] Hilera González, J.; Martínez Hernando, v. (1995). Redes neuronales artificiales: fundamentos, modelos y aplicaciones. Madrid: RA-MA.

[Holland 1975] Holland, J. H. (1975). Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. U Michigan Press.

[Kirkpatrick 1983] Kirkpatrick, S., Jr., D. G., & Vecchi, M. P. (1983). Optimization by simmulated annealing. science, 220(4598), 671-680.

[Koza,1992] Koza, J. R. (1992). Genetic programming: On the Programming of Computers by Means of natural Selection (Complex Adaptative Systems). The MIT Press.

[Larrosa 2003] González, P. M., & Larrosa, J. (2003). Restricciones Blanda: Modelos y Algoritmos. Inteligencia artificial: Revista Iberoamericana de Inteligencia Artificial, 7(20), 69-82

[Pose 2000] Pose, M. G. (2000). Introducción a los Algoritmos Genéticos. Departamento de Tecnologías de la Información y las Comunicaciones Universidad de Coruña.

[Regueiro 1995] REGUEIRO, C; ef. al. (1995). «Modelos básicos de redes neuronales artificiales». En: BARRO, S.; MIRA, J. (eds.) Computación

neuronal. Santiago de Compostela: Universidade, Servicio de Publicacions e Intercambio Científico. (Cursos e Congresos da Universidade de Santiago de Compostela, 86).

[Schaerf 1995] Schaerf, A. (1995). A survey of automated timetabling.

[Schaerf 1996] Schaerf, A. (1996). Tabu search techniques for large high-school timetabling problems. Computer Science, Department of Interactive Systems, CWI.

[Spears 1993] Spears, W. M., De Jong, K. A., Bäck, T., Fogel, D. B., & De Garis, H. (1993, January). An overview of evolutionary computation. In Machine Learning: ECML-93 (pp. 442-459). Springer Berlin Heidelberg.

[Tallavó 1999] Marcos Gil Tallavó, Amadís Antonio Martínez, "Algoritmo basado en Tabú search para el problema de asignación de horarios de clases", 1999. Recuperado de: http://servicio.bc.uc.edu.ve/facyt/v1n1/1-1-8.pdf Visitado el 20 de abril de 2013

[Van Laarhoven 1987] Van Laarhoven, P. J., & Aarts, E. H. (1987). Simulated annealing (pp. 7-15). Springer Netherlands.

[Wren 1996] Wren, A. (1996). Scheduling, timetabling and rostering—a special relationship? In Practice and theory of automated timetabling (pp. 46-75). Springer Berlin Heidelberg.