Universidad Tecnológica de la Mixteca

Implementación de un Algoritmo Genético para la Distribución Óptima de Horarios de Clases en CONALEP

Tesis que para obtener el título de:

INGENIERO EN COMPUTACION

Presenta:
Miguel Ángel Herrera Soriano

Dirigida por:
M. en C. Agustin Santiago Alvarado

Acatlima, Huajuapan de León, Oaxaca. Febrero del 2000

DEDICATORIAS

Dedico este trabajo a mi familia, en especial a mi hijo Miguel Ángel Herrera Mendoza.

Siempre serás especial...

AGRADECIMIENTOS

Al M. en C. Agustín Santiago Alvarado por el apoyo, sugerencias y haber aceptado dirigir este proyecto.

Al Ing. Alejandro Enrique por sus útiles sugerencias y comentarios durante la realización de este proyecto.

A mi mamá, papá, hermanos, les doy las gracias por todo ese apoyo en todos los aspectos, por supuesto sin olvidar a mi hermana Clara que tuvo confianza en mi y gracias a su insistencia y apoyo ha sido posible realizar esta tesis.

Finalmente a mi esposa Ana Mendoza y a mi hijo Miguel Ángel, les agradezco su apoyo y paciencia que me han brindado para poder concluir este trabajo.

Índice

Capítulo 3. Condiciones y características del problema

Capítulo 4. Implementación del algoritmo

Capítulo 5. Pruebas y resultados

PLANTEAMIENTO DEL PROBLEMA.

En las escuelas de educación media superior la asignación de horarios de clase se ha convertido en una actividad compleja y tediosa, principalmente porque se espera obtener el óptimo aprovechamiento de los recursos con los que cuenta la institución, ya que la población estudiantil ha rebasado la infraestructura disponible, en su elaboración intervienen factores como: salones y laboratorios de cómputo, la cantidad de materias a asignar, así como la cantidad necesaria de profesores para impartir clases.

En algunas ocasiones esta tarea no se realiza satisfactoriamente, presentándose los siguientes resultados:

- Solapamiento de horas clase al asignar a un grupo salones distintos a la misma hora o al asignar un profesor a grupos distintos a la misma hora.
- Pérdida de tiempo debido al extravió de tablas de horarios de clase ya elaborados o al buscar y consultar los datos de una determinada tabla de horarios.

Al observar estos contratiempos, nos hemos dado cuenta que el personal del plantel CONALEP 145 no cuenta con un procedimiento que garantice la óptima distribución de horarios de clase a los grupos formados y la asignación de los cursos a los profesores.

Por lo que nos planteamos el siguiente problema:

¿Qué grado de eficiencia presenta el procedimiento que siguen las instituciones educativas en la elaboración de horarios de clase de grupos, laboratorios de cómputo y profesores?

En la actualidad las computadoras tienen una capacidad de cómputo que no se puede despreciar, la cual se incrementa cada vez más y permite ejecutar algoritmos más eficaces y complejos en poco tiempo.

Se propone un programa para computadora que realice la asignación de horarios de clase en una institución como el CONALEP tomando en cuenta las restricciones presentadas incrementando la eficiencia en el procedimiento que sigue la institución para la elaboración de horarios, obteniendo a su vez el óptimo aprovechamiento de los recursos de la institución y la pronta realización de dicha actividad en favor de un mejor servicio al alumnado y plantel docente en general.

INTRODUCCION

En la actualidad, la computadora es una herramienta poderosa que se ha convertido en el medio más eficaz para llevar acabo diversas actividades complejas y rutinarias del ser humano, como lo es la asignación de horarios de clase en las instituciones educativas como el CONALEP (Colegio Nacional de Educación Profesional Técnica).

Desde su creación, el CONALEP no presentaba grandes conflictos y problemas para la asignación de horarios de clase, ya que el número de grupos y salones disponibles era equitativo. En la actualidad el número de grupos es mayor que la cantidad de salones disponibles, por lo que se tiene que prestar especial atención en dicha asignación de horarios para aprovechar al máximo los salones, laboratorios y planta docente y no ocasionar gastos extras y pérdida de tiempo al realizar una asignación de horarios equivocada.

Una asignación de horarios de clase sin la debida planeación provoca serios conflictos al momento de colocar las materias en los grupos correspondientes, ya que en algunas ocasiones se asignan a un mismo grupo dos clases a la misma hora o a algunos profesores se les asigna impartir dos clases a la misma hora. Actualmente este proceso se realiza de manera manual por el personal administrativo, los cuales se auxilian con una computadora para copiar las tablas e imprimirlas.

El no aprovechar la gran capacidad de procesamiento de datos de una computadora hace que este tipo de problemas sean complejos y largos en tiempo para el ser humano. Quizás al principio el elaborar una serie de instrucciones para ser ejecutadas por la computadora sea una tarea difícil y tediosa, pero una vez realizadas, los resultados serán inmediatos en tiempo, fácil de realizar y básicamente el problema será resuelto, permitiendo en lo futuro ahorrar tiempo y dinero a la institución.

En este trabajo se elaboró un programa que permitirá realizar la asignación óptima de horarios de clase, atendiendo a las necesidades que se presentan en el CONALEP, el cual trabajará utilizando la técnica de algoritmos genéticos en la fase de optimización, los cuales se basan en la teoría darwiniana de la evolución de los seres vivos, que aplica los procedimientos como la reproducción, la mutación, adaptación, extinción y selección natural. Los algoritmos genéticos imitan estos procedimientos por medio de rutinas o programas para computadora, simulando los procesos naturales y utilizando la velocidad de procesamiento de una computadora para solucionar problemas con una gran cantidad de posibles soluciones que se deben analizar para obtener la óptima solución al problema en cuestión.

El contenido de este trabajo se divide de la siguiente manera:

En el capítulo 1 se realiza una revisión de las técnicas de optimización (búsqueda global) y se resaltan las características y deficiencias de cada una de ellas. A continuación en el capítulo 2 se describe la técnica de los algoritmos genéticos y se muestra el contenido del algoritmo, más adelante en el capítulo 3 se describe el problema a resolver, sus

características y condiciones, en el capítulo 4 se muestran las rutinas utilizadas en la realización del programa para computadora y se describe cada una de ellas, en el capítulo 5 se presentan las pruebas y resultados obtenidos con el algoritmo implementado, a continuación se presentan las conclusiones de este trabajo, finalmente se presentan las referencias y los apéndices, los cuales contienen el código del programa, las tablas de horarios que se elaboran en el plantel y el manual del usuario.

CAPÍTULO 1

MÉTODOS DE OPTIMIZACIÓN

1.1. Introducción

La asignación de horarios de clases es un problema con N variables, las cuales representan los parámetros a obtener como materia, turno, semestre, carrera, día, hora, etc. a partir de una función de evaluación. La asignación de un horario determinado a una materia, determina la solución final del problema

En los procesos tradicionales de optimización se selecciona un valor inicial, que representa una asignación de materias en una hora determinada y con la aplicación de un algoritmo de optimización, avanza en el conjunto de posibles soluciones para encontrar una asignación para una materia que sea aceptable (es lo que se conoce como punto de partida) y al continuar las asignaciones, el programa debe evaluar en alguna forma que dicha asignación es satisfactoria y aceptada para así llegar a la solución final. Cuando esto no sucede, el programa se detiene o comienzan nuevamente, encontrando lo que se conoce como óptimo local, estos métodos tradicionales de optimización utilizan rutinas basadas en el descenso o ascenso sistemático (según se plantee la función de evaluación), esto quiere decir que sólo se aceptan aquellos pasos que decrecen o aumentan el valor aceptado por la función de evaluación, provocando que el ascenso o descenso se detenga en un mínimo o máximo local.

Con estas características se determina que la solución que se obtiene depende del punto de inicio del proceso. Los métodos convencionales trabajan exitosamente y convergen rápidamente a una buena solución cuando el punto inicial es bueno, en caso contrario la solución alcanzada no será la mejor. Por ello, se presenta una revisión de los métodos tradicionales de optimización para conocer sus ventajas y desventajas, y en que clase de problemas resultan eficientes, dando soluciones satisfactorias.

1.2. Métodos tradicionales de optimización y búsqueda

La robustez de un algoritmo es el balance entre la eficiencia y la eficacia necesaria para funcionar en diferentes ambientes. Esto quiere decir, ¿Qué tan buena es la solución encontrada? y ¿Qué tan rápido encontró esa solución?. Mientras mayor sea la robustez de un sistema, el costo de rediseño se ve reducido o eliminado.

A continuación se presenta un estudio que es importante para la pregunta acerca de si los métodos de búsqueda tradicionales reúnen la eficiencia y eficacia necesaria en la búsqueda de soluciones.

En la actualidad los métodos de búsqueda se clasifican en tres tipos principales: métodos basados en cálculo, métodos numéricos y métodos aleatorios[1]. Examinaremos cada uno de ellos para observar que características tienen y bajo que condiciones arrojan resultados satisfactorios.

1.2.1. Métodos basados en el cálculo

Los métodos basados en el cálculo son los métodos que más ampliamente se han estudiado. Estos se clasifican en dos clases: Indirectos y Directos.

Los métodos indirectos buscan un extremo local resolviendo un conjunto de ecuaciones no lineales, que resulta del gradiente de la función de merito igualada a cero[2]. Por ejemplo dado un plano y la gráfica de una función como la que se muestra en la figura 1a), para encontrar un posible pico empieza por restringir la búsqueda en estos puntos con inclinaciones a cero en todos los sentidos.

Por otra parte, los métodos directos buscan el óptimo local pero en base a la función de merito y moviéndose en una dirección relacionada al gradiente local, esto es, la noción del seguimiento del gradiente (hill-climbing)[3], para encontrar el mejor punto local la función asciende en la mejor dirección posible. Ambos métodos han sido mejorados, aumentados, o reducidos, pero aún así no dan resultados completamente satisfactorios, algunas de las razones de su falta de robustez son:

Primero, ambos métodos son de alcance local. Suponga que la figura 1a) representa una porción del dominio completo; una figura completa de la gráfica se presenta en la figura. 1b), en la cual se observa que al iniciar la búsqueda de una solución o un óptimo en la proximidad del pico más corto, causaría perder el evento principal (el pico más alto). Una vez que se ha alcanzado el pico más corto, una posible mejora es reiniciar aleatoriamente el proceso de búsqueda o algún otro truco para salir de éste y encontrar un pico mejor, lo que no hace de manera automática, ya que no hay un criterio que le diga cuando parar la búsqueda.

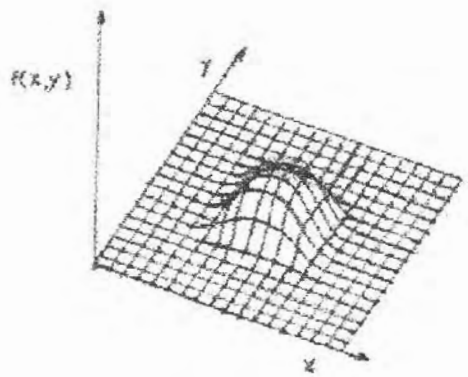


Figura. 1a). *Porción del dominio de una función f(x,y).*

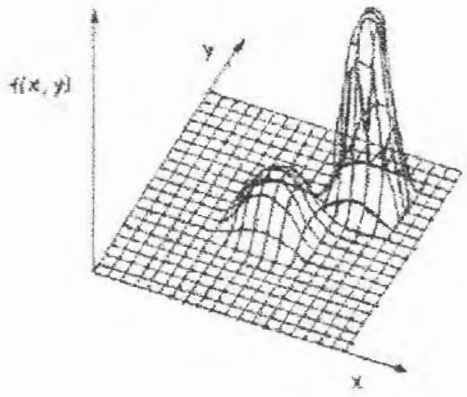


Figura 1b). *Dominio completo de la función f(x,y)*

Segundo, los métodos basados en el cálculo dependen de la existencia de derivadas de orden superior (Valores de inclinación bien definidos). Aún cuando se permita la aproximación numérica de derivadas, esto es un severo defecto, porque al existir pequeños

errores de redondeo o errores en el cálculo de las derivadas se obtendrán evaluaciones peores que la inicial, siendo una situación poco deseada.

En el mundo real de búsqueda el conjunto de solución a problemas es discontinuo y multimodo, como el que se muestra en la figura 1.2 y para los métodos que dependen fuertemente de la continuidad y multiderivadas, esto se vuelve una gran restricción para su aplicación, por lo que son utilizados en un caso muy limitado de problemas. Por esta razón y debido a su inherente alcance local de búsqueda, debemos rechazar los métodos basados en cálculo. No son robustos en dominios extensos.

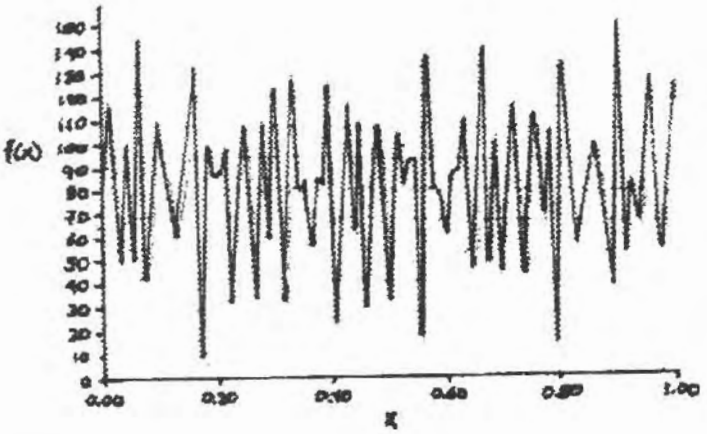


Figura 1.2. *Función con gran cantidad de máximos y mínimos.*

1.2.1.1. Métodos matemáticos

Se fundamentan en el cálculo diferencial para establecer estrategias que mejoren sistemas a partir de un sistema inicial. Estos métodos utilizan sólo las primeras derivadas porque el tiempo de cómputo y el trabajo involucrado no permiten el cálculo de derivadas de alto orden porque las variables que se evalúan en segundas o terceras derivadas provocan errores de redondeo o fallas al calcular las derivadas, además de que se requiere de mayor tiempo de cómputo para calcular una segunda o tercera derivada.

En estos casos se presenta la siguiente función de merito:

$$f_i = f_{0i} + \sum_{j=1}^{N} a_{ij}(x_j - x_{0j}) \qquad \text{para i} = 1,2,3,...,M \tag{1.1}$$

Donde f_i es la función a evaluar en los diferentes puntos del espacio de búsqueda, f_{0i} es el valor de f_i en el punto X_0 del espacio de búsqueda, formado por los valores x_{0j} que representan el punto de inicio al comenzar el proceso de optimización.

En notación matricial

$$F = F_0 + A(X - X_0).$$

Si se considera a $fi = 0$ para todo i, se obtiene un sistema de ecuaciones simultaneas en $x_j - x_{0j}$, y al resolverlo se obtienen los cambios en las N variables que interviene en un sistema cuyas ecuaciones se igualan a cero, obteniendo la ecuación

$$A(X - X_0) = -F_0 \tag{1.2}$$

Este método tiene el inconveniente de que la matriz A debe ser cuadrada para mantener el balance del sistema, es decir tantas variables como ecuaciones deben forman el sistema. Las soluciones obtenidas de la ecuación (1.1) en la mayoría resulta peor que la anterior.

1.2.1.2. Mínimos cuadrados

Una propuesta diferente a la anterior es minimizar los valores de las f_i en lugar de igualarlas con cero. Para esto se toman las siguientes ecuaciones

$$\phi = \sum_{i=1}^{M} f_i^2 \ , \tag{1.3}$$

donde ϕ es la función de evaluación, derivando a ϕ con respecto a cada x_j tenemos

$$\frac{\partial \phi}{\partial x_i} = \sum_{i=1}^{M} 2 f_i a_{ik} \ ; \quad para\ k = 1, 2, ..., N, \tag{1.4}$$

$$a_{ij} = \frac{\partial f_i}{\partial x_j} \tag{1.5}$$

sustituyendo f_i de la ecuación (1.1) e igualando con cero, se obtienen las siguientes N ecuaciones en $x_j - x_{0j}$,

$$\sum_{i=1}^{M} a_{ik} f_{0i} + \sum_{i=1}^{M} \sum_{j=1}^{N} a_{ij} a_{ik} (x_j - x_{0j}) = 0, \tag{1.6}$$

en notación matricial,

$$A^T A(X - X_0) + A^T F_0 = 0 \, . \tag{1.7}$$

Las ecuaciones (1.6) y (1.7) son las ecuaciones clásicas de mínimos cuadrados, estas no necesitan igual número de variables y de ecuaciones.

La solución a la ecuación (1.7) es de la forma:

$$X - X_0 = -(A^T A)^{-1} A^T F_0 \tag{1.8}$$

Al existir pequeños errores de redondeo o errores en el cálculo de las derivadas, afectarán la solución que se obtenga, aún cuando la exactitud fuera infinita, la no linealidad de F provoca que el óptimo que se obtenga será sólo en el caso lineal y esto provocará obtener un sistema peor que el inicial cuando el punto de partida es malo.

1.2.2. Métodos numéricos

Los métodos numéricos han sido considerados en muchas formas y tamaños. La idea es ir justamente en línea recta; con un espacio de búsqueda finito o un espacio de búsqueda infinito discreto, el algoritmo inicia observando los valores de la función objetivo en cada punto en el espacio, uno a la vez. Aunque la simplicidad de este tipo de algoritmo es atractiva, tales esquemas carecen de robustez por una simple razón: carecen de eficiencia por la cantidad de tiempo que se requiere para evaluar espacios muy grandes punto por punto y tener todavía una oportunidad de usar la información para algo práctico[4]. Cada vez menos esquemas numéricos son usados para problemas reales.

1.2.3. Métodos de búsqueda aleatoria

Los algoritmos de búsqueda aleatoria han logrado incrementar su popularidad debido a que los investigadores han reconocido las carencias de los métodos basados en cálculo y de los esquemas numéricos. Sin embargo, los esquemas aleatorios que buscan y salvan el mejor punto evaluado como una posible solución del problema que se estudia, podrían ser discontinuos debido a los requerimientos de eficiencia. Las búsquedas aleatorias a la larga pueden no mejorar a los esquemas numéricos. Sin embargo se deben distinguir los métodos de búsqueda estrictamente aleatoria de las técnicas aleatorias, de la siguiente manera: un método aleatorio es aquel que avanza en el conjunto de puntos al azar, es decir, si existe una mejoría en la evaluación de un punto continua en ese rumbo, en caso contrario cambia su rumbo a cualquier otro punto elegido al azar. Las técnicas aleatorias son aquellas que sirven como herramienta a algún proceso de búsqueda, por ejemplo: los algoritmos genéticos son uno de los proceso de búsqueda que usa elección aleatoria como una herramienta para guiar la búsqueda.

Usar elección aleatoria como una herramienta en un proceso de búsqueda dirigida es extraño al principio, pero la naturaleza contiene muchos ejemplos. Otra técnica popular de búsqueda aleatoria es el llamado recocido simulado se ayuda del proceso aleatorio para guiar su formas de búsqueda de estados de energía mínimos que representan las posibles soluciones del problema. Se debe aclarar que una búsqueda aleatoria no necesariamente implica una búsqueda sin dirección.

1.2.3.1. Recocido simulado

Este proceso consiste en obtener estados de baja energía en un sólido, exponiéndolo a un calentamiento. Físicamente el recocido primeramente reblandece el sólido calentándolo a una temperatura elevada, para luego enfriarlo lentamente, esto provoca que las partículas del sólido vuelvan al estado fundamental del sólido por si mismas. En cada temperatura del proceso las partículas del sólido alcanzarán un equilibrio térmico si el enfriamiento se

produce lentamente; por otra parte, si el enfriamiento se produce muy rápido, dichas partículas podrán llegar a estados meta-estables y no a su estado fundamental en donde las partículas toman una posición perfecta y el sistema esta en su nivel más bajo de energía, en cambio en los estados meta-estables se localizan defectos formando estructuras de alta energía.

1.2.3.1.1 Algoritmo Metrópolis

El algoritmo de metrópolis[5] simula el proceso de un baño térmico en un sólido y obtiene el equilibrio térmico a una determinada temperatura generando un número elevado de transiciones, utilizando la distribución de Boltzmann[6] para describir el equilibrio térmico.

El algoritmo realiza el paso de un estado de energía a otro siguiendo estas reglas: si el estado generado posee una energía menor que el estado que se tiene actualmente, entonces se acepta el estado generado como el estado actual; en caso contrario, el estado generado se aceptará con una probabilidad determinada por la distribución de Boltzmann. Esta función depende de la temperatura y de la diferencia entre los dos niveles de energía. Mientras menor sea la temperatura, menor probabilidad habrá de obtener estados de mayor energía y mientras mayor sea la energía del nuevo estado, menor probabilidad tendrá de ser aceptado, esto quiere decir que cada estado tiene una posibilidad de ser alcanzado, pero con diferente probabilidad a diferente temperatura.

Los estados que forman el sistema corresponden a las soluciones del problema; la energía de los estados con la evaluación de calidad de cada solución; el estado fundamental con la solución óptima y los estados meta-estables corresponden a los óptimos locales. La temperatura desempeña un papel de control.

Este algoritmo simula el funcionamiento del recocido simulado, a continuación se presentan los pasos que se siguen para llevarse a cabo.

1. Configuración inicial, se elige Ci, aleatoriamente o por un método heurístico. A esta configuración se asocia un costo, $Ei = F(Ci)$.

2. Temperatura inicial, se determina T.

3. Generar un cambio aleatorio en la configuración. Este cambio posiblemente producirá una nueva configuración, $Ci+1$.

4. Si $Ei+1 < Ei$, entonces el cambio se aceptará siempre.

5. Si $Ei+1 > Ei$, entonces el cambio se acepta según la probabilidad igual a $exp[-(Ei+1 - Ei)/kT]$

6. Actualizar la temperatura T si se requiere. Regresar al paso 2.

Repetir el ciclo hasta terminar.

1.3. Técnicas de Inteligencia Artificial (IA).

La inteligencia artificial se compone de varios campos como: sistemas expertos, robótica, el procesamiento del lenguaje natural entre otras. En la mayoría de estas aplicaciones el punto central es la solución de problemas. La IA tiene el objetivo de crear un resolutor general de problemas, un programa que pueda proporcionar soluciones a los diferentes tipos de problemas en los que no se tiene un conocimiento bien diseñado[7]. En los inicios de la IA, su objetivo fue encontrar buenos métodos de búsqueda, porque el obstáculo principal es la magnitud y la complejidad de las situaciones a los que se aplican estas técnicas. La IA considera que la búsqueda es el centro de la solución de problemas, que es un componente importante de la inteligencia.

Por lo general, existen dos tipos de problemas. El primero se puede resolver mediante el uso de procedimientos deterministas en los que se garantiza el éxito (efectuando un cálculo), los métodos que se aplican en este tipo de problemas son traducidos en algoritmos que fácilmente se implementan en programas que la computadora puede ejecutar. El segundo tipo de problemas son aquellos que no se pueden resolver de manera determinista. En el mundo real existen pocos problemas que se pueden resolver de esta manera, la mayoría no son problemas computacionales y estos se resuelven mediante la búsqueda de una solución (la IA interviene aplicando un método de resolución de problemas).

El siguiente ejemplo ilustra la representación y terminología en la búsqueda aplicada por IA. Imagine que ha perdido las llaves de su coche y lo que recuerda es que están en un lugar de su casa, distribuida de la siguiente manera (ver figura 1.3).

Cuando comienza a buscar, usted se localiza en la entrada (X), busca en el comedor, a continuación busca en el recibidor, al dormitorio1, al dormitorio2, regresa al recibidor y pasa al dormitorio principal. Al no encontrar nada, regresa al comedor y encuentra sus llaves en la cocina, esta búsqueda se puede representar en un grafo como el de la figura 1.4.

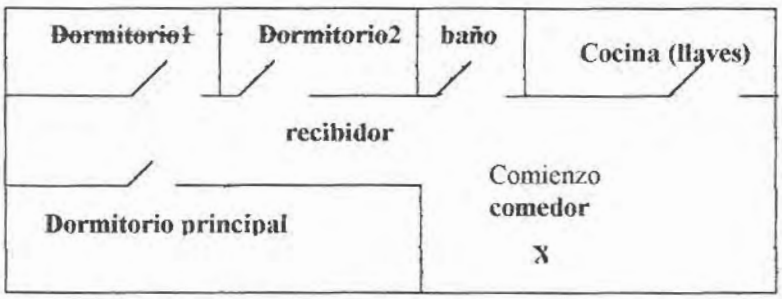


Figura 1.3. *Esquema que representa la distribución de los cuartos en la casa.*

Un grafo representa la forma de trabajar de una técnica de búsqueda. Se manejan los siguientes conceptos:

Nodo	Un punto discreto y posible objetivo.
Nodo terminal	Un nodo que es el final de un camino.
Espacio de búsqueda	El conjunto de todos los nodos
Objetivo	El nodo que es el objeto de la búsqueda
Heuristica	Información de que un nodo sea la mejor opción que otro.
Camino solución	Grafo dirigido de los nodos que conducen a la solución.

En el ejemplo de las llaves, cada habitación es un nodo; el espacio de búsqueda esta representado por toda la casa; el objetivo es la cocina y el camino solución se muestra en la figura 1.4. Los nodos terminales son los dormitorios, la cocina y el baño porque ya no dejan ir más allá.

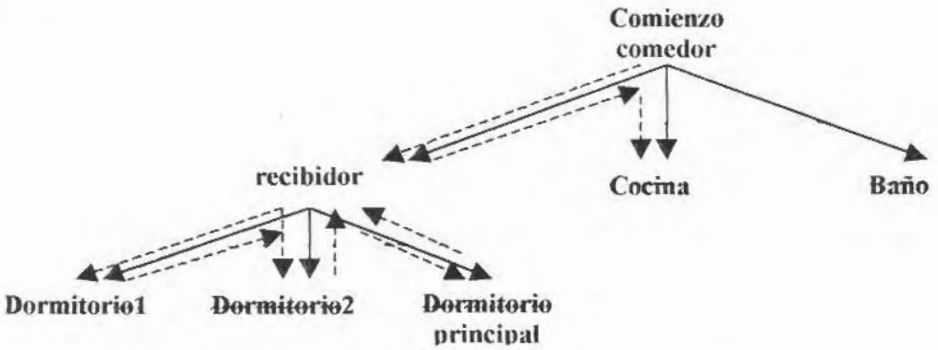


Figura 1.4. *Grafo de las llaves perdidas.*

1.3.1. Explosión combinatoria

En el problema de las llaves, la búsqueda de la solución es relativamente fácil, se inicia en un punto de la casa y se busca de cualquier forma hasta llegar a la solución, pero esta situación no se encuentra en la mayoría de los problemas. Generalmente se utiliza una computadora para resolver problemas con un número de nodos en el espacio de búsqueda muy grande que crece en la medida que crecen los posibles caminos hacia el objetivo. La situación que se presenta es que al añadir un nuevo nodo, se añade más de un camino.

Si se deseara acomodar en todas las formas posibles tres objetos A, B, C, se tendría el siguiente resultado:

A B C
A C B
B C A
B A C
C B A
C A B

Estas son las únicas seis formas de colocar los objetos, si los objetos fueran 4, habría 24 combinaciones posibles, con 5 se tendrían 120, en el caso de 1000 objetos el número de combinaciones es extremadamente grande.

La explosión combinatoria se refiere a tener una gran cantidad de posibilidades y en poco tiempo se hace imposible examinar todas ellas. Al tener nodos adicionales en el espacio de búsqueda se incrementan las posibles soluciones en más de uno, y en determinado momento se tendrán demasiadas posibilidades. Con estas características, sólo a los problemas más simples se les aplica una búsqueda exhaustiva, es decir examinar todos los nodos del espacio de búsqueda, lo que se conoce como la técnica de la "fuerza bruta".

La técnica de la "fuerza bruta" funciona siempre, pero no es práctica, porque consume demasiado tiempo y recursos de computación, para esto se han desarrollado nuevas técnicas de búsqueda.

1.3.1.1. Técnicas de búsqueda

Las técnicas de búsqueda más comunes son:

- Búsqueda de primero en profundidad.
- Búsqueda de primero en anchura
- Búsqueda del menor costo
- Búsqueda de la escalada.

En la evaluación de una búsqueda intervienen dos medidas importantes:

- ¿Qué tan rápido encuentra la solución?
- ¿Qué tan buena es la solución?

La diferencia que existe en encontrar una solución óptima y encontrar una buena solución es que la óptima requiere una búsqueda exhaustiva que indique que ya no es posible encontrar una mejor solución que la que se tiene. Para encontrar una buena solución implica encontrar una solución que satisfaga determinadas restricciones, sin importar que exista una mejor solución.

La rapidez en la búsqueda esta determinada por la longitud del camino solución y por los nodos recorridos. Algunas técnicas de búsqueda funcionan mejor en unas situaciones que en otras. La forma en que se define el problema ayuda en cierta medida a elegir el método de búsqueda apropiado.

1.3.1.1.1. La búsqueda primero en profundidad

Analiza cada camino hasta su conclusión antes de analizar cualquier otro camino.

En la siguiente figura 1.5, F es la solución:

Esta búsqueda recorre el grafo en el siguiente orden: ABDBEBACF.

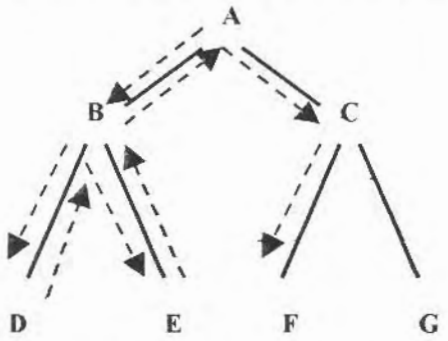


Figura 1.5. *Búsqueda en profundidad.*

Esta técnica es útil para localizar el objetivo, en el peor de los casos se convierte en una búsqueda exhaustiva. El rendimiento puede ser bastante bajo en los casos en los que se analiza una rama larga sin encontrar solución al final o en ella, consumiendo un tiempo considerable al analizar los nodos y en el regreso para analizar otra rama.

1.3.1.1.2. La búsqueda primero en anchura

Examina cada nodo de un mismo nivel antes de continuar al siguiente nivel de profundidad. En la figura 1.6, C es la solución:

Se visitan los nodos ABC, esta búsqueda garantiza una solución, en caso de existir, pues degenera en una búsqueda exhaustiva. El resultado que se obtiene de esta búsqueda depende de la organización que se tenga en la información al momento de guardarla en la computadora. Esta búsqueda realiza un esfuerzo sustancial en el caso en que la solución se encuentre a varios niveles de profundidad.

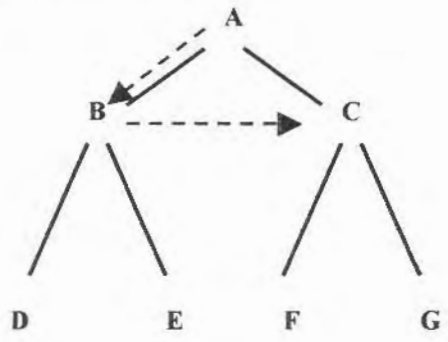


Figura 1.6. *Búsqueda en anchura*

Las búsquedas descritas se realizan a ciegas, las búsquedas que realizan lo hacen sin ninguna información extra por parte de la computadora, este proceder funciona en ciertos problemas, la manera de mejorar la búsqueda es agregando heurísticas.

La información heurística no es precisa ni garantizada, pero aumenta las posibilidades de que un método de búsqueda logre su objetivo rápidamente[8].

1.3.1.1.3 La búsqueda de la escalada

Esta búsqueda elige como siguiente paso el nodo que pareciera estar más próximo a la solución. El nombre de escalada proviene del ejemplo de un escalador perdido, donde lo único que sabe es que su campamento se encuentra en la cima de la montaña, de tal forma de que cada paso que asciende lo hace en dirección correcta.

Al aplicar la búsqueda de la escalada se presentan tres situaciones:

1. Los picos falsos.
   Cuando se analiza y asciende en una ruta larga sin encontrar solución y se debe regresar en el mismo camino recorrido para analizar otra opción u otra ruta.

2. Las mesetas.
   Cuando los pasos son igual de buenos o malos (presentan el mismo valor), convirtiendo en una búsqueda en profundidad.

3. La cumbre.
   Cuando se realizan grandes avances sin llegar a la solución y debido al regreso se repite el recorrido.

La escalada tiene mayor probabilidad de obtener una solución en comparación con los métodos no heurísticos.

1.3.1.1.4. La búsqueda del menor costo.

Es contraria a la escalada, elige el camino de menor esfuerzo. A diferencia de la escalada que minimiza las conexiones, la búsqueda del menor costo minimiza la distancia a recorrer. Esta búsqueda puede encontrar falsos valles, falsas tierras bajas y falsos desfiladeros, pero en promedio presenta un buen rendimiento.

1.4. Conclusiones

Se analizaron diferentes métodos y técnicas de búsqueda, demostrándose que funcionan bien si el punto de inicio de la búsqueda es apropiado para encontrar el punto óptimo, en caso contrario conducen a óptimos locales teniendo una fuerte dependencia del punto inicial.

La deficiencia de los métodos tradicionales al trabajar con funciones no lineales se debe al hecho de no incluir derivadas de orden superior, porque el tiempo de computo y el trabajo que se requiere para calcular una segunda o tercera derivada es mayor, además de

que al aplicar derivadas superiores se obtienen errores de redondeo o errores en el cálculo de la derivada.

Cando el número de variables se incrementa, el espacio de solución se vuelve tan grande que imposibilita encontrar el óptimo global.

CAPÍTULO 2

ALGORITMOS GENÉTICOS

2.1. Introducción

Las características de los seres vivos como la evolución y la selección natural les ha proporcionado habilidades y destrezas para soluciónar sus problemas, esto nos muestra una capacidad tal que avergüenza a quienes se dedican a resolver problemas, que dedican meses o años de trabajo intelectual para obtener una solución.

Estas cualidades de la evolución hay que imitarlas, los algoritmos genéticos fueron inventados para imitar algunos procesos observados en la evolución natural, para resolver problemas de la misma manera que lo hace la naturaleza.

En la teoría de la evolución existen mecanismos que aún no están claramente entendidos, pero otros mecanismos si se conocen, tales como los cromosomas que son dispositivos orgánicos que codifican la estructura de los seres vivos lo que permite su evolución. Parte de la creación de un individuo se encuentra en el proceso de codificación y decodificación del cromosoma, esta parte del proceso de la evolución no está totalmente comprendida, pero se han encontrado otras características del proceso que han sido ampliamente aceptadas.

1) El proceso de la evolución opera sobre los cromosomas mas no sobre los seres vivos que ellos codifican.

2) La selección natural determina la supervivencia de los individuos de la población, permite que las estructura aptas se reproduzcan con mayor frecuencia que las estructuras poco aptas, es decir, la selección natural es una relación entre los cromosomas y la capacidad de adaptarse de sus estructuras decodificadas. En este proceso, si un organismo falla en alguna prueba de adaptación no tiene más opción que morir.

3) El proceso de reproducción garantiza la mezcla y combinación de los genes de los individuos que componen la población y originar nuevos individuos. En la unión del óvulo y el espermatozoide los cromosomas homólogos se entrecruzan en zonas intermedias lo que permite el intercambio de material genético. Al realizarse esta mezcla y cruzamiento, la evolución de los seres vivos es más rápido que si se originaran copias exactas de cada individuo modificados solamente por una mutación del cromosoma.

4) La única información con la que se cuenta en este proceso de evolución es la contenida en el conjunto de cromosomas que representa cada individuo y en la estructura que decodifica estos cromosomas.

Jonh Holland[9] un investigador de computación, en los años 70's incorpora estas características de la evolución en un programa para computadora, llegando a una técnica que permita resolver problemas complejos de la misma forma que la naturaleza lo hace a

través del proceso de la evolución. Inició sus trabajos con algoritmos que manipulaban dígitos binarios (unos y ceros) representados por medio de cadenas que llamó cromosomas. imitando a la naturaleza, él manipulaba a ciegas estos cromosomas tratando de encontrar cada vez mejores cromosomas. Estos algoritmos no saben nada acerca del problema a resolver, la única información que se les proporcionó fue la evaluación de cada cromosoma que se producía durante el proceso, esta evaluación se usa sólo para la selección de cromosomas, tal es así que los cromosomas con las mejores evaluaciones tendían a reproducirse en mayor número que los cromosomas con mala evaluación.

Holland llamó a esta técnica algoritmos genéticos porque se basan en los procesos genéticos de los seres vivos[10].

2.2 Panorama general del Algoritmo Genético

Un algoritmo genético presenta dos mecanismos importantes que lo relacionan con el problema que se pretende resolver, uno es la codificación del problema en un cromosoma y el otro es la función de evaluación la cual da una medida del valor de los cromosomas en el contexto del problema. Esta última es esencial en los procesos de optimización.

La codificación del problema varía según el problema. Holland manejaba cadenas de bits, pero ésta no es la única forma de codificar un problema, es muy probable que no haya una técnica que funcione bien en todos los problemas por ello es parte importante del éxito de la aplicación del algoritmo la selección de una buena técnica de codificación y la elección de la función de evaluación la cual es la conexión entre el algoritmo genético y el problema que se va a resolver. Una función de evaluación tiene como entrada a un cromosoma y como salida un número o una lista de miembros que es una medida del funcionamiento del cromosoma en el problema a ser resuelto. La función de evaluación juega el mismo papel en el algoritmo genético que el medio ambiente en la evolución natural. La interacción de un individuo con su ambiente proporciona una medida de su aptitud para sobrevivir, y la interacción de un cromosoma con la función de evaluación proporciona una medida de aptitud que el algoritmo genético usa cuando lleva a cabo la etapa de reproducción.

En este proceso que simula la evolución, una población inicial de cromosomas no aptos mejorará cuando los padres sean remplazados por mejores y mejores hijos. El mejor individuo en la población final puede ser una solución altamente evolucionada para el problema a resolver.

2.2.1. Descripción de un algoritmo genético

Un algoritmo genético

1. Genera una población inicial de cromosomas.
2. Evalúa cada cromosoma de la población
3. Crea nuevos cromosomas apareando los cromosomas actuales.
4. Borra miembros de la población para hacerle un lugar a los nuevos cromosomas.

5. Evalúa los nuevos cromosomas y los inserta en la población.
6. Si el tiempo se ha terminado, se detiene y da como resultado el mejor cromosoma, si no, va al inciso 3.

El diagrama de flujo que representa el funcionamiento de un algoritmo genético es el siguiente:

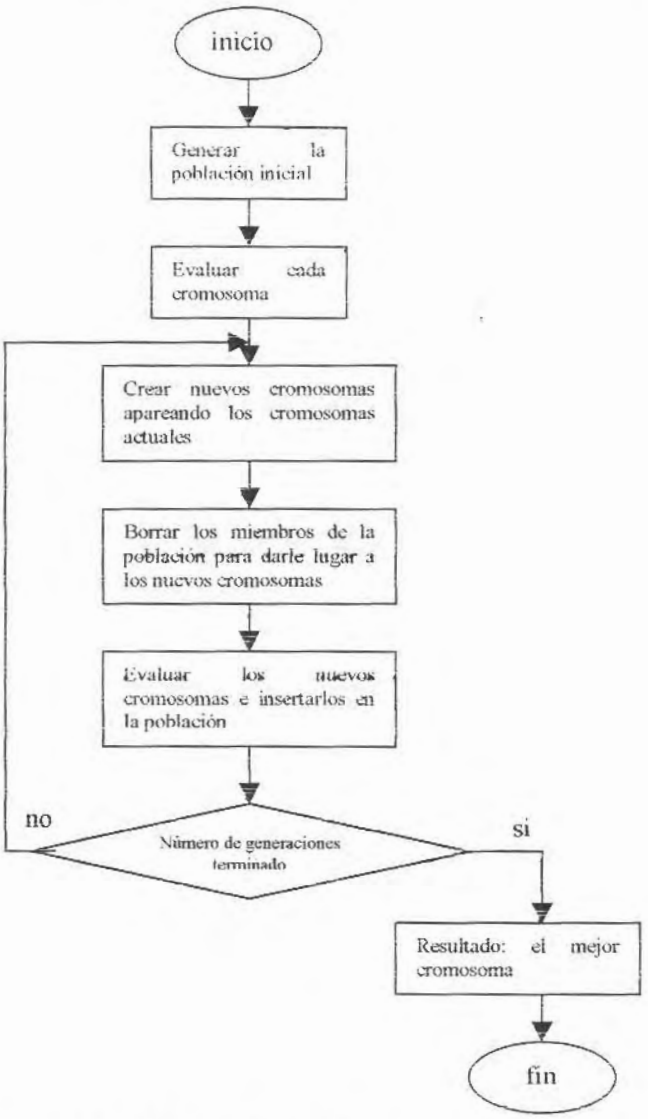


Diagrama de flujo 2.1. *Funcionamiento de un algoritmo genético*

2.3. Diferencias entre los algoritmos genéticos y los métodos tradicionales de optimización

Los algoritmos genéticos difieren de los métodos tradicionales en:

1) Los algoritmos genéticos trabajan con un código de los parámetros, y no con los parámetros mismos. Los parámetros del problema a optimizar se codifican en cadenas finitas sobre un alfabeto finito.

2) Los algoritmos genéticos utilizan poblaciones de puntos y no puntos individuales. En los métodos de optimización tradicionales se usan reglas de transición deterministas para mover la búsqueda de un punto a otro en el espacio de soluciones, este procedimiento es la forma perfecta de localizar óptimos falsos en un espacio que contenga múltiples máximos y mínimos. Por su parte, los algoritmos genéticos utilizan una base de datos de puntos, esto les permite explorar simultáneamente muchas colinas en paralelo, de tal forma que la probabilidad de encontrar óptimos falsos se reduce enormemente.

3) Los algoritmos genéticos usan la información de la función de evaluación y no sus derivadas u otro conocimiento auxiliar. Las técnicas de búsqueda tradicionales requieren mucha información auxiliar para trabajar apropiadamente. Por ejemplo, las técnicas del gradiente necesitan derivadas (calculadas analítica o numéricamente) para ser capaces de encontrar los máximos o los mínimos. En cambio, los algoritmos genéticos no necesitan esa información auxiliar, sólo requieren los valores de la función de evaluación asociados con las cadenas individuales.

4) Los algoritmos genéticos usan reglas de transición probabilísticas, en lugar de reglas deterministas. El uso de probabilidades no significa que el método sea una búsqueda aleatoria simple, los algoritmos genéticos usan la aleatoriedad como una herramienta para guiar la búsqueda hacia regiones con mejores probabilidades de encontrar una solución.

2.4. La anatomía de un algoritmo genético

El procedimiento de mayor complejidad dentro de la anatomía de un algoritmo genético es copiar cadenas e intercambiar cadenas parcialmente.

Un algoritmo genético está compuesto por:

1. Un generador de números aleatorios.
2. Una base de datos.
3. Un módulo para reproducción, uno para cruzas, uno para mutaciones y uno de evaluación.
4. El módulo principal.

2.4.1 Generador de números aleatorios

Los lenguajes de alto nivel generalmente proporcionan entre sus rutinas un generador de números aleatorios. La distribución obtenida es uniforme porque cada número del rango tiene la misma probabilidad de ocurrir. Usando el generador de números aleatorios se define la función flip que funciona de manera similar a cuando se lanza una moneda, esta función retorna un 1 o un $0^{[11]}$.

En esta función, la variable probabilidad contiene un valor de 0.5

```
Función flip (probabilidad)
        a = un número dado al azar por la computadora
        si a <= probabilidad entonces regresa 1
        en otro caso
                si  a > probabilidad entonces regresa 0
        fin otro caso
fin de función
```

2.4.2. Estructura de datos

El Algoritmo genético genera y procesa poblaciones de cadenas. Por tanto, se construye la población como un arreglo de individuos (Pob[][]), donde cada individuo contiene el fenotipo (los parámetros descodificados), el genotipo (la cadena de bit) y la evaluación (función de evaluación). [12].

El tamaño máximo de la población y la longitud del cromosoma son parámetros que se definen al inicio de la ejecución de un programa basado en algoritmos genéticos.

```
función popinicial ()
        desde i = 1 hasta tamaño máximo de la población
                desde j = 1 hasta longitud máxima del cromosoma
                        Pob[i][j] = flip(0.5)
                incrementa j
        incrementa i
fin de función
```

2.4.3. Selección de cromosomas

El siguiente paso en el proceso es, seleccionar los cromosomas que se van a reproducir y posteriormente a cruzar para obtener los cromosomas que van a formar la siguiente generación. El propósito de seleccionar algunas cadenas es para darles más oportunidades de reproducirse a los miembros de la población que son más aptos o que tienen las mayores evaluaciones. Hay muchas maneras de hacer la selección, una de ellas es la técnica conocida como *la rueda de la ruleta*[13].

En el siguiente ejemplo se muestra una población de 10 cromosomas junto con sus evaluaciones, la suma total de las evaluaciones es 76. El primer renglón contiene el número del cromosoma, el segundo renglón contiene las evaluaciones y el tercero la suma acumulativa de las evaluaciones. En la tabla 2.1 se muestra también siete números, entre 0 y 76, generados aleatoriamente, junto con el número del cromosoma que ha sido seleccionado por la técnica de la rueda de la ruleta, en cada caso el cromosoma seleccionado es el primero para el cual la suma acumulativa de evaluaciones es mayor o igual al número aleatorio.

Cromosoma	1	2	3	4	5	6	7	8	9	10
Evaluación	8	2	17	7	2	12	11	7	3	7
Suma acumulativa	8	10	27	34	36	48	59	66	69	76
Número aleatorio			23	49	76	13	1	27	57	
Cromosoma seleccionado			3	7	10	3	1	3	7	

Tabla 2.1. *Ejemplo de diez cromosomas junto con sus evaluaciones aplicando la rueda de la ruleta.*

El efecto de la selección con la rueda de la ruleta es obtener una cadena seleccionada aleatoriamente. Aunque este procedimiento de selección es aleatorio, la oportunidad que tiene cada cadena de ser seleccionada es directamente proporcional a su evaluación. Al transcurrir las generaciones, este mecanismo elimina los miembros con menores evaluaciones y tiende a expandir el material genético de los miembros mejor evaluados. Por supuesto que es posible que el peor miembro de la población sea seleccionado cada vez que se use el procedimiento, pero la probabilidad de que esto suceda en una población de cualquier tamaño es despreciable.

Este procedimiento de selección es llamado de la rueda porque es equivalente a asignarle una rebanada de un círculo a cada miembro de la población, de tal forma que el tamaño de la rebanada sea proporcional a su evaluación, después de esto se gira el círculo y se lanza un dardo, la cadena seleccionada es aquella donde cayo el dardo.

La función de selección que representa la técnica de la rueda de la ruleta es la siguiente[14]:

```
Función selección ()
        sumpar = 0
        i = 0
        rand = (un número aleatorio entre 0 y 1) * (sf)
        hacer
                i = i +1
                sumpar =  sumpar + f[i]
        hasta sumpar >= rand or i = tpoblación
        regresa i
fin de función
```

donde sf es la suma total de evaluaciones y f[i] es la evaluación de la cadena i, sumpar es la suma parcial de los valores de evaluación f[i], tpoblación es el número máximo de individuos existentes en la población.

2.4.4. Cruza de cromosomas

El apareamiento o cruza de dos individuos provoca el intercambio de material genético que se considera como el motor del proceso de evolución. En un algoritmo genético una cruza mezcla el material genético de dos cromosomas para crear dos hijos o descendientes. Holland propuso un operador de cruzas que llamó *cruza en un punto*[15]. Este operador

funciona mediante un punto de corte seleccionado aleatoriamente en los dos cromosomas participantes, intercambiando las dos partes del cromosoma después de este punto.

Este operador produce hijos que son totalmente diferentes de sus padres, como lo muestra la figura 2.1. Otra característica es que no introduce diferencias en un bit en una posición donde ambos padres tengan el mismo valor, en el segundo ejemplo, obsérvese que en las posiciones 2, 3, 4 y 5 los padres tienen el mismo valor al igual que los hijos. Un caso extremo ocurre cuando ambos padres son idénticos, en este caso la cruza no introduce diversidad en los hijos.

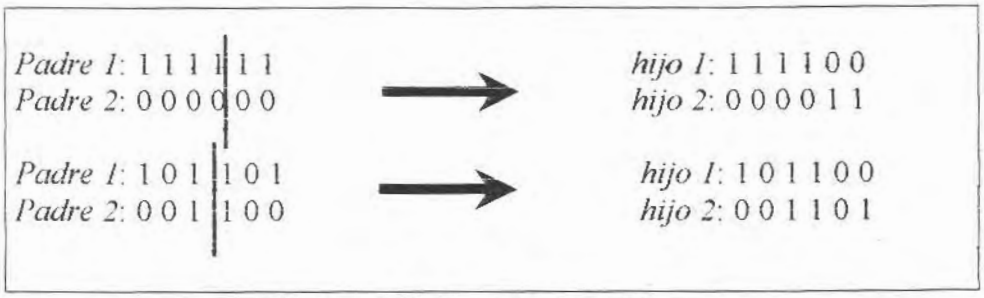


Figura 2.1. *Cruza de cromosomas*

La cruza es un elemento importante en un algoritmo genético, sin este operador los algoritmos genéticos no funcionan. El funcionamiento de este operador es la característica que distingue a los algoritmos genéticos de otros algoritmos. La rutina que realiza la cruza de cromosomas es la siguiente[16]:

```
función cruza (i1, conyuge1, conyuge2)
jcruza = aleatorio (1, lcromosoma - 1)
        ncruza = ncruza + 1
        para j = 1 hasta jcruza
              hijo1[i1,j] = pob[conyuge1, j]
              hijo2[i1,j] = pob[conyuge2, j]
        incrementa j
        si jcruza <> lcromosoma entonces
             para j = jcruza hasta lcromosoma
                   hijo1[i1, j] = pob[conyuge2, j]
                   hijo2[i1, j] = pob[cónyuge 1, j]
             incrementa j
fin de función
```

donde i1 es el valor de la fila en la que se colocarán los valores en la matriz que guarda a los nuevos individuos, conyuge2 y conyuge1 indican que individuos de la población se cruzarán, jcruza es el valor que marca cuantos bits de la cadena que compone a cada individuo se modifican y cuantos no se modifican, ncruza indica cuantas veces se ha realizado el cruzamiento de individuos, lcromosoma es valor que indica la longitud del cromosoma.

En esta subrutina aparece la función aleatorio, esta proporciona números aleatorios en un intervalo definido por dos números llamados menor y mayor. La función aleatorio se muestra a continuación[17].

```
función aleatorio (menor, mayor)
        a = random((mayor - menor + 1) + menor)
        Sí a > mayor entonces a = mayor
        Sí menor >= mayor entonces a = menor
regresa a
fin de función
```

donde menor y mayor indican los límites en los cuales se obtendrá un número aleatorio.

2.4.5. La mutación

La mutación es una característica más del proceso de evolución, mantiene la diversidad al impedir que una población se vuelva homogénea y permite que el proceso de evolución continúe.

Esta característica se realiza de la siguiente manera:

- Selecciona aleatoriamente un cromosoma de la población.
- Selecciona aleatoriamente un gene del cromosoma.
- Cambiar el valor del gene seleccionado.

La aplicación de este procedimiento se realiza cuando las características de los miembros de la población son muy similares en todos ellos. A continuación se muestra la rutina para llevar a cabo la mutación en una población[18].

```
función mutación (población ())
        10: imuta = selección()
        Sí imuta = valmax entoces goto 10
        jmuta = aleatorio(1, lcromosoma)
        Sí población[imuta][jmuta] = 1 entonces
                población[imuta][jmuta] = 0
        en otro caso
                Sí población[imuta][jmuta] = 0 entonces
                        población[imuta][jmuta] = 1
fin de función
```

donde selección() se refiere a la función que se encarga de determinar a que individuo se le va a aplicar la mutación, valmax es el valor a obtener o a alcanzar por el algoritmo genético, una vez alcanzado este valor, el programa se detiene.

2.4.6. La extinción

Esta característica trabaja de la siguiente manera:

- Se impone al algoritmo un número determinado de mutaciones que pueden levarse acabo en determinada población.
- Si después de haberse aplicado este número de mutaciones no se ha alcanzado el objetivo deseado, se localiza el mejor individuo hasta ese momento con el fin de mantener información de la población anterior.
- Se borra la población, es decir se borran los datos de todos los individuos de la base de datos.
- Se genera una nueva población.

Este procedimiento es similar a la extinción de los dinosaurios que se extinguieron al no poder adaptarse a las condiciones del medio ambiente, dando paso a otras especies mejor adaptadas.

2.5 Esquemas o bloques construidos.

La búsqueda de soluciones óptimas para un problema utilizando algoritmos genéticos consiste en buscar determinadas cadenas binarias. Esto se puede imaginar como un paisaje donde se localizan montañas o cimas y valles, estas montañas representan las cadenas más óptimas y los valles las soluciones menos deseadas o menos aptas. Mientras más elevada sea la cima más óptima será la solución.

Estas cadenas binarias también nos indican la posición en determinadas regiones del espacio de soluciones, esto se obtiene observando el valor que posee la cadena en determinados lugares, es decir, todas las cadenas que empiezan con uno representan una región en el espacio de soluciones, así como las cadenas que empiezan con cero o las cadenas que poseen un uno en la cuarta posición o un cero en la última, etc.

La técnica de la escalada nos permite explorar el espacio de soluciones, esta comienza en un punto al azar, si la calidad de la solución se mejora aún ligeramente se continúa en esa dirección, en caso contrario se toma la dirección opuesta. En los problemas complejos se presenta un paisaje con muchas cimas y valles. La determinación del sentido que seguirá la búsqueda se torna más complicada debido a que tales espacios de búsqueda son enormes.

Los algoritmos genéticos exploran este paisaje de la siguiente manera, imaginemos que se echa una red sobre este paisaje. La variedad de cadenas que forman la población sondea muchas regiones a la vez. Esto nos lleva a obtener muestras en diferentes regiones con la probabilidad de hallar una buena solución en ese entorno.

El algoritmo genético explota las regiones de más alto rendimiento del espacio de soluciones porque al aplicar en las sucesivas generaciones la reproducción y cruza generan un mayor número de cadenas que pertenecen a dichas regiones. Se considera que el número de cadenas de una región aumenta según la estimación estadística de la idoneidad de esa región. Un estadístico tendría que evaluar docenas de muestras tomadas de millones de regiones para determinar la idoneidad media de cada región. El algoritmo genético alcanza el mismo resultado con menos cadenas y prácticamente sin cómputo alguno.

La clave de esta conducta sorprendente, reside en que cada cadena pertenece a todas las regiones en las cuales aparece cualquiera de sus bits. La cadena 11011001 pertenece a las regiones 11****** (donde * indica que es indiferente el valor del bit correspondiente), 1******1, **0**00* entre otras. A estas regiones se les llama *bloques o esquemas construidos*[19]. Las regiones que contienen bits sin especificar, serán agrupadas por una fracción grande de todas las cadenas de la población. Un algoritmo genético que manipula una población de cientos de cadenas, en realidad toma muestras de un número de regiones mucho mayor. Tal paralelismo proporciona al algoritmo genético su ventaja principal sobre otros procedimientos.

Para concretar lo anterior, consideremos que las cadenas están construidas sobre un alfabeto binario $V = \{0,1\}$, las letras mayúsculas representan cadenas y sus componentes se indicarán con minúsculas. Por ejemplo una cadena de 7 bits puede representarse como

$$A = a_1a_2a_3a_4a_5a_6a_7 , \tag{2.1}$$

donde A es la cadena o cromosoma, los a_i son los genes y los valores que puede tomar a_i son los alelos.

Una población se denota como $A(t)$ y se forma, por las cadenas A_j, $j = 1, 2, \ldots, n$, existentes en el tiempo t. Un esquema H esta definido sobre un alfabeto de tres símbolos $V = \{0, 1, *\}$. Por ejemplo, un esquema de longitud 7 puede ser 111*0**, note que la cadena $A = 0\ 111000$ pertenece al esquema H.

Una cadena binaria de longitud l tiene 3^l esquemas. En una población con n miembros existen como máximo $3^l n$ esquemas. Estos datos nos dan una idea de la cantidad de información que procesa un algoritmo genético.

2.5.1. Orden de un esquema

El orden de un esquema se denota por $o(H)$, es el número de posiciones fijas. Por ejemplo, el esquema $H = *11\ *0**$ tiene un orden igual a 3, es decir, $o(H) = 3$.

La longitud de un esquema, denotada por $\delta(H)$, es la distancia entre la primera y última posición especificada en la cadena, en el ejemplo anterior $\delta(H) = 3$.

Determinar el efecto de la reproducción sobre el número esperado de esquemas de una población se determina de la siguiente manera. Imagine que en un tiempo dado t hay m ejemplos de un esquema H que pertenecen a una población $A(t)$, denotándose como:

$$m=m(H,t) \tag{2.2}$$

Al efectuarse la reproducción, una cadena se copia según el valor de su evaluación. Esto quiere decir que una cadena A_j se selecciona según la probabilidad:

$$p_j = \frac{f_j}{\Sigma f_j} \tag{2.3}$$

donde f_j es la evaluación de A_j Por lo tanto, en el tiempo t + 1

$$m(H, t+1) = m(H, t)\frac{f(H)}{f^-} \tag{2.4}$$

donde $f(H)$ es la evaluación promedio de las cadenas que contienen al esquema H en el tiempo t, y

$$f^- = \frac{\Sigma fi}{n} \tag{2.5}$$

El crecimiento de un esquema es proporcional al crecimiento del cociente de las evaluaciones promedio del esquema y el promedio de las evaluaciones de la población. Es decir, *esquemas con evaluaciones por arriba del promedio de la población incrementan su número en la próxima generación*, mientras que esquemas con evaluaciones por debajo del promedio de la población decrecen en número. Este funcionamiento se lleva acabo con todos los esquemas contenidos en una población [20].

Suponiendo que un esquema H permanece por arriba del promedio en una cantidad cf, donde c es una constante, entonces

$$m(H, t+1) = m(H, t)\frac{(f^- + cf^-)}{f^-} \tag{2.6}$$

o bien : $$m(H, t+1) = (1 + c)m(H, t) \tag{2.7}$$

Si el proceso se inicia en $t = 0$ y se supone un valor estacionario de c, la ecuación anterior se convierte en:

$$m(H,t) = m(H,0)(1+c)^t , \tag{2.8}$$

Esta última ecuación nos indica una progresión geométrica. El efecto de la reproducción nos indica que el número de cadenas que contienen a un esquema que esté por encima (o por debajo) del promedio crecen (o disminuyen) exponencialmente. El procedimiento de cruza afecta a las cadenas de diferente manera, se tiene una cadena de 7 genes y dos de sus esquemas (ver figura 2.2).

$$\begin{array}{lll} A = & 0\ 1\ 1 & 1\ 0\ 0\ 0 \\ H1 = & *\ 1\ * & *\ *\ *\ 0 \\ H2 = & *\ *\ * & 1\ 0\ *\ * \end{array}$$

Figura 2.2 *Proceso de cruza en dos esquemas*

En el proceso de cruza se selecciona aleatoriamente un punto de cruza y después se intercambian las subcadenas. Supongamos que el punto de cruza es en el lugar 3 como indica la línea vertical, se observa que el procedimiento de cruza afecta de manera distinta a los esquemas, el esquema *H1* es destruido mientras que el esquema *H2* sobrevive, nótese que la probabilidad de que un esquema sobreviva depende de la longitud del esquema.

Observemos que *&(H1)*=5, si el punto de cruza se selecciona al azar pero con una distribución uniforme, entonces el esquema se destruye con una probabilidad

$$p_d = \frac{\delta(H1)}{l-1} \tag{2.9}$$

donde *l* es la longitud de la cadena, es decir $p_d = 5/6$ y de la misma manera para el esquema *H2*.

Suponga que el proceso de cruzamiento se lleva a cabo con una selección aleatoria, digamos con probabilidad p_c, entonces la probabilidad de que un esquema sobreviva estará dada como

$$p_s >= 1 - p_c \frac{\delta(H)}{l-1}, \tag{2.10}$$

y suponiendo que los procesos de reproducción y cruza son independientes, tenemos

$$m(H, t+1) >= m(H,t)\frac{f(H)}{f^-}\left[1-p_c\frac{\delta(H)}{l-1}\right] \tag{2.11}$$

Del efecto que tienen los procedimientos de cruza y reproducción sobre un esquema dependerá si dicho esquema crece o desaparece, esto depende de un factor multiplicativo y depende de dos cosas: una es que el esquema se encuentre por arriba o por debajo del promedio de la población y otro es la longitud corta o larga del esquema. Entonces aquellos esquemas que estén por arriba del promedio y que tengan longitudes cortas se incrementaran exponencialmente. A esta conclusión se le ha dado el nombre de *teorema de los esquemas o teorema fundamental de los algoritmos genéticos*[21].

2.6. Conclusiones

Se explicó en que consisten los algoritmos genéticos así como su funcionamiento y sus rutinas más importantes.

Se describieron las ventajas en eficiencia y eficacia de este algoritmo sobre los métodos tradicionales de optimización.

Se expuso el resultado más importante de estos algoritmos que es el teorema fundamental de los algoritmos genéticos, el cual indica el funcionamiento de los algoritmos genéticos con un paralelismo implícito.

CAPÍTULO 3

CONDICIONES Y CARACTERÍSTICAS DEL PROBLEMA

3.1. Introducción

Las escuelas de educación media superior en la actualidad dedican especial atención a la realización de horarios de clase, esta actividad implica un análisis de la infraestructura con la que cuenta la institución, como son aulas disponibles como salones de clase, laboratorios de cómputo y se determinará el número de profesores necesarios para impartir clases a los grupos formados.

En instituciones como el CONALEP (Colegio Nacional de Educación Profesional Técnica), es importante dar el mejor uso a los laboratorios de cómputo, ya que esta institución ofrece una capacitación en informática y cerca del 70% de horas clase de cada materia de esta área son horas prácticas, es decir requieren el uso de una computadora. Esta información junto con la cantidad de salones disponibles, permitirá realizar la formación de los grupos en las diferentes carreras y semestres.

Para el primer semestre, la cantidad de alumnos de nuevo ingreso que se aceptan depende de la cantidad de salones disponibles. En la mayoría de las escuelas, el número de aulas casi siempre es menor a la población estudiantil, por lo que se requiere un uso óptimo de la infraestructura disponible para brindar una enseñanza apropiada a los estudiantes.

En este capítulo presentamos de manera detallada la problemática que se presenta al realizar la asignación de horarios.

3.2. Condiciones y Características

La asignación de horarios de clases en un colegio es una actividad semestral y su complejidad varía según la cantidad de grupos que integran el plantel, así como su infraestructura disponible, la cantidad de materias a asignar a cada grupo, de cada una de las diferentes carreras y de cada semestre.

El número de grupos a formar depende de la demanda de los estudiantes para ingresar a cada una de las diferentes carreras que ofrece la institución, de ellas, la más solicitada es la carrera de informática, seguida por la carrera de contabilidad y por último la carrera de administración.

La carrera de informática cuenta con más alumnos, lo que dará como consecuencia la formación de mayor número de grupos que las otras carreras. Para cumplir con el objetivo de esta carrera, se debe contar también con equipo de cómputo útil, actualizado y suficiente para cubrir las necesidades de los alumnos, lo que a su vez limita la formación de grupos.

El plantel CONALEP cuenta con dos laboratorios de cómputo con una capacidad de 20 computadoras cada uno, están disponibles en un horario de 7 de la mañana a 9 de la noche, con este equipo y con esta disponibilidad de horario se alcanzan a cubrir las necesidades que se tienen en esta carrera por el momento, para esto los laboratorios de cómputo se encuentran ocupados en su máxima capacidad durante el horario disponible.

Las materias a asignar en cada grupo presentan las siguientes características:

Se distinguen dos grandes áreas en las que se agrupan las materias: el área de formación básica y el área de formación ocupacional.

El área de formación básica agrupa las materias que se imparten a todos los grupos de un mismo semestre sin distinción de la carrera a la que pertenezcan dichos grupos.

El área de formación ocupacional agrupa materias que son exclusivas para cada carrera, considerando el semestre al que pertenece, de tal forma que se tendrán materias de primer semestre que sólo se impartirán a los grupos que correspondan a la carrera para la cual se destina cada materia.

La duración de cada carrera en la institución CONALEP es de seis semestres, estos semestres se encuentran por periodos de la siguiente manera: de septiembre a febrero se imparten clases a los semestres de primero, tercero y quinto. En el período de marzo a agosto se imparten clases a segundo, cuarto y sexto semestre. De esta manera se tienen periodos con semestres pares o impares, en ninguna ocasión se han combinado estos periodos.

Una característica más de las materias a asignar es el número de horas clase a la semana y por si fuera poco, la cantidad de horas clase es diferente para cada materia, existen materias con tres horas clase a la semana y materias con dieciocho horas clase a la semana.

Al momento de asignar horarios de clase a cada grupo, se deben asignar preferencias a las materias, dependiendo del tipo que estas sean. Por lo regular las materias que requieren el uso de los laboratorios de cómputo se les trata de asignar en primer lugar y a continuación las materias con horas teóricas o que no requieren el uso del laboratorio.

Otra característica son las horas clase teóricas y horas clase prácticas. Las horas clase teóricas se refiere a tomar la clase en un salón de clase común, es decir en un salón en el que se disponga de gis y pizarrón y no requiera del uso de computadoras.

Las horas clase prácticas son aquellas que se asignan o se toman en un laboratorio de cómputo, porque estas implican la realización de prácticas y se requiere para ello el uso de una computadora.

Con la realización del análisis y la obtención de información de cuáles y cuántas materias se impartirán a los grupos del próximo período escolar, el siguiente paso es la

elaboración de tablas de horarios por grupo, esta actividad presenta las siguientes características:

Todo el proceso es manual, sólo se cuenta con listas de materias, listas de grupos, listas de profesores, lápiz y papel para las gráficas que representarán cada grupo.

Se utiliza una computadora sólo para la captura e impresión de las tablas con materias ya asignadas.

Para la realización de este proceso se dedican dos semanas con horario de trabajo completo y en ocasiones horas extras para obtener una buena distribución de horarios.

Las personas que realizan esta actividad deben contar con el conocimiento acerca de los recursos disponibles y los niveles de preferencia e importancia de las materias, grupos y profesores, además de disponer de material y equipo necesarios para realizar la elaboración de horarios.

3.2.1. Material y equipo necesarios

El personal encargado para esta actividad debe reunir el siguiente material:

- Lista de materias a asignar según el período que corresponda a la actual distribución.

Esta lista se ordena por turno, semestre y carrera e incluso se ordena por el tipo de materia. Esto último se refiere a materias con horas teóricas y prácticas o sólo con horas teóricas.

- Lista de grupos que ya se han integrado.

La información que presenta esta lista por cada grupo es la siguiente:

- Turno
- Semestre
- Carrera
- Grupo

Estos datos integran un número de identificación de cada grupo, de tal forma que al leerlo, se conoce el semestre, el turno, la carrera y el número de grupo consecutivo de la misma carrera.

- Lista de profesores.

Esta lista contiene los siguientes datos:

- Nombre del profesor.
- Profesión o área a la que pertenece.

- Tablas en papel, en las que se irá registrando la materia, el día y la hora según corresponda en la distribución de materias.

La forma de la tabla es la siguiente (ver tabla 3.1):

CARRERA: **SEMESTRE:** **TURNO:** **GRUPO:**

HORA	LUNES	MARTES	MIERCOLES	JUEVES	VIERNES
7:00 - 8:00					
8:00 - 9:00					
9:00 - 10:00					
10:00 - 11:00					
11:00 - 12:00					
12:00 - 13:00					
13:00 - 14:00					

Tabla 3.1. *Tabla de horarios de clase del CONALEP*

En la columna izquierda de la tabla se marca la hora, que dependerá del turno que corresponda para el grupo que en ese momento se estén asignando sus materias. La parte superior indica los días en los que se impartirán clases. Cada espacio de la cuadrícula es un lugar disponible para colocar las horas clase de la materia que se asigne.

El personal debe contar con tablas exclusivas para cada laboratorio de cómputo, en las cuales se indicará el registro de horas ya asignadas a los diferentes grupos en los laboratorios de cómputo. Cada tabla representa un laboratorio e indica el turno al que pertenece. En estas tablas a diferencia de las que representa cada grupo, se anotan los números de los grupos a los que a sido asignadas las horas de laboratorio, en las tablas de grupos se registra el nombre de la materia que se está asignando.

El equipo necesario para esta actividad es una computadora que será útil para la captura de las tablas ya asignadas e imprimirlas para su publicación en el colegio. Los programas para computadora que utilizan son un procesador de palabras u hoja de cálculo. Una vez que se ha reunido el material y equipo necesario se establecen las preferencias y condiciones para las materias, grupos y profesores, esto permitirá saber que materias se asignarán antes que el resto, también que grupos tendrán preferencias al asignar un salón o a un determinado profesor.

La descripción del procedimiento de asignación de horarios de clase que se hace a continuación corresponde a un proceso que considera los aspectos más generales posibles para llevarse a cabo.

Paso uno:

De la lista de materias seleccionar la que corresponda en el orden de asignación. Cada materia presenta la siguiente información:

- Nombre de la materia

- Horas clase prácticas a la semana
- Carrera a la que pertenece.

Paso dos:

De la lista de grupos tomar el que corresponda con el semestre y la carrera que indica la materia a asignar. Cada grupo presenta la siguiente información:

- Turno
- Semestre
- Carrera
- Grupo

Si la característica carrera de los datos de la materia indica que es una materia de formación básica, entonces se asignará a todos los grupos que correspondan con el semestre que indica la materia sin importar la carrera a la que corresponda cada uno de ellos. En caso contrario, si la materia es de formación ocupacional, esta sólo se asignará a los grupos que correspondan con el semestre y carrera que indiquen los datos de la materia.

Paso tres:

Tomar una hoja que contenga una tabla disponible y rotularla con la información del grupo, esto se deberá hacer en el caso de que sea la primera materia asignar en un determinado grupo, de no ser así, se deberá localizar la hoja que corresponda con el grupo al que le corresponde la asignación de la materia.

Paso cuatro:

Una vez rotulada o localizada la hoja correspondiente, se elige el día y la hora en la que se asignará cada hora clase de la materia, hasta finalizar con todas las horas clase correspondientes a la materia. La hora de asignación dependerá del turno al que corresponda el grupo y al profesor que se asigne.

Paso cinco:

Revisar la lista de grupos y observar si existen más grupos para los cuales corresponda la materia que se esta asignando, en caso de haberlos, elegir uno y repetir los pasos dos, tres y cuatro, hasta que ya no existan grupos en la lista que deban contener esta materia en sus horarios.

Paso seis:

Una vez finalizada la asignación de esta materia, se descarta de la lista y se toma la próxima a asignar y se repiten los pasos uno, dos, tres, cuatro y cinco hasta finalizar la lista de materias. Teniendo presente que si un profesor da el mismo curso varias veces, estos no queden solapados.

Una vez finalizada la asignación de esta materia, se descarta de la lista y se toma la próxima a asignar y se repiten los pasos tres, cuatro y cinco hasta finalizar la lista de materias. Teniendo presente que si un profesor da el mismo curso varias veces, estos no queden solapados.

3.2.2. Resultados del procedimiento

Algunos de los problemas que se presentan al realizar la asignación de horarios de clase con este procedimiento son los siguientes:

- El solapamiento de materias, grupos y horas es un caso que suele presentarse.
- Una distribución de horarios de laboratorio de cómputo equivocada repercute en los horarios de cada grupo relacionado con los laboratorios y viceversa
- Confusión al manejar hojas de papel que tienen las listas de materias y grupos.
- Perdida de listas o tablas.
- Perdida de tiempo en la búsqueda de materias, grupos o tablas.
- Saber elegir la opción óptima de una gran cantidad de opciones disponibles para colocar una materia.
- Desacuerdo por parte del profesor a su horario asignado.

Estas son algunas consecuencias de la asignación de horarios de clase, debido a esto, las tablas de horario se modifican en varias ocasiones hasta obtener una distribución deseada y aceptable, tal es la complejidad de esta actividad que una vez iniciadas las clases, algunas veces existen confusiones en los horarios, lo que implica una revisión extra, provocando trabajo extra por parte del personal encargado para reacomodar los horarios.

Un factor más que influye fuertemente en la distribución de horarios es la disponibilidad de los profesores que impartirán clase en el período que se inicia. Esta disponibilidad se refiere a las horas que los profesores dedican para impartir clase, ya que por política de la institución, los profesores que ahí laboran no son de tiempo completo, aportando sólo una horas de su tiempo para impartir clases. Esta disponibilidad provoca que la asignación de horarios este sujeta al horario que cada profesor destina a impartir clases. Para minimizar el problema que causa esta disponibilidad de los profesores, los horarios de cada profesor se han agrupado en dos turnos, aquellos profesores que tienen disponibilidad por la mañana, se les asigna el turno matutino y aquellos cuya disponibilidad es por la tarde, se les asigna el turno vespertino, esta medida ha tenido éxito y el problema se ha reducido grandemente, permitiendo una distribución menos compleja de las materias en los horarios de clase para cada grupo.

También es importante mencionar que la institución asigna calificaciones a cada profesor en función de su nivel académico, así como la participación y colaboración en las actividades de la institución y su desempeño como profesor en el período que termina. Esta evaluación que se realiza de los profesores, será un indicador para la asignación de horas de trabajo en el siguiente período. Cabe mencionar que cada profesor tendrá como máximo veinte horas de trabajo a la semana en la institución.

Debido a que esta actividad resulta ser difícil y tediosa de realizar para el personal administrativo, se realizó un programa para computadora que lleva a cabo el procedimiento de asignación de horarios de clases, obteniendo con esto más utilidad de un recurso como la computadora, reduciendo el tiempo que se dedica normalmente a esta actividad así como la seguridad en el manejo de la información, permitiendo al personal administrativo realizar sus actividades con más comodidad.

En páginas posteriores se hace una descripción del programa para computadora que se implementó, el cual utiliza la técnica de algoritmos genéticos para la asignación de horarios de clase.

3.3. Conclusiones

La asignación de horarios de clase que realiza el personal administrativo del plantel CONALEP, es hecha de manera manual, lo que ocasiona el dedicar mayor tiempo a esta actividad, afectando la realización de otras actividades administrativas.

Las personas encargadas deben realizar un análisis de la situación del plantel para saber la capacidad con la que se cuenta y poder brindar un servicio educativo adecuado a los estudiantes.

El procedimiento que realiza el personal administrativo además de ser tedioso y largo en tiempo, muchas veces no resulta ser el más favorable, por lo que en varias ocasiones se deben realizar modificaciones a los horarios de clase.

CAPÍTULO 4

IMPLEMENTACIÓN DEL ALGORITMO.

4.1. Introducción

La aplicación de la técnica de algoritmos genéticos ha tenido gran éxito en los problemas de optimización, en los que se a observado que son eficientes y confiables. Aún así, no todos los problemas son apropiados para aplicar esta técnica, por lo que es necesario considerar las ***características*** del mismo antes de utilizar la técnica.

1. Su espacio de búsqueda debe estar limitado dentro de un rango.

2. Debe poderse definir una función de aptitud que indique el nivel de aceptación de una cierta respuesta.

3. Las soluciones deben codificarse de una forma que su implementación en la computadora resulte ser fácil.

La primer característica se refiere a la aplicación de la técnica a problemas con espacios de búsqueda discretos, aunque también se intenta aplicar a problemas con espacios de búsqueda continuos, pero con un rango de soluciones pequeño.

La función de aptitud es la función objetivo del problema a optimizar, una característica de la función de aptitud es que debe ser capaz de valorar a los individuos que presenten una mejor solución y de rechazar a las peores soluciones, con el fin de que las mejores se propaguen con mayor rapidez.

La codificación de soluciones generalmente es a través de cadenas binarias, formados por unos y ceros, en algunos casos se utilizan números reales y letras. Para el problema que se tiene en particular, esta técnica permite obtener soluciones aceptables, en un tiempo relativamente corto y garantizando resultados altamente funcionales

4.2. Alcances y limitaciones del algoritmo implementado

El programa que a continuación se describe realiza la asignación de horarios a cada grupo, tomando en cuenta los siguientes parámetros: el turno, la carrera, el semestre y por supuesto las materias que correspondan con dichos parámetros. La asignación que se realiza esta ligada a dos condiciones más, una es la disponibilidad de horario del profesor y otra la disponibilidad de horario en los centros de cómputo.

Este programa muestra las tablas de horarios de cada grupo, así como de cada laboratorio y de cada profesor como resultado de la aplicación de un algoritmo genético al problema. Al mostrarse los resultados en pantalla, el usuario será capaz de consultar una nueva tabla de algún grupo que aparece en la lista, podrá hacer modificaciones en las asignaciones de cada materia en la tabla mostrada, esto sólo lo podrá hacer en las tablas

correspondientes a los grupos, si la modificación es válida el programa cambia la asignación, en caso contrario, el programa pone un mensaje indicando que la asignación no está disponible, existen dos razones para esto, una es que la nueva hora ya esté asignada en los laboratorios de cómputo y la otra es que en esa hora el profesor no tenga disponibilidad.

El usuario podrá también imprimir la tabla que en ese momento está consultando, con la aclaración de que la impresora debe estar conectada al equipo de cómputo, encendida y con papel suficiente, el programa no detecta estos errores de impresión, si no se detecta la impresora, el programa seguirá trabajando de manera normal.

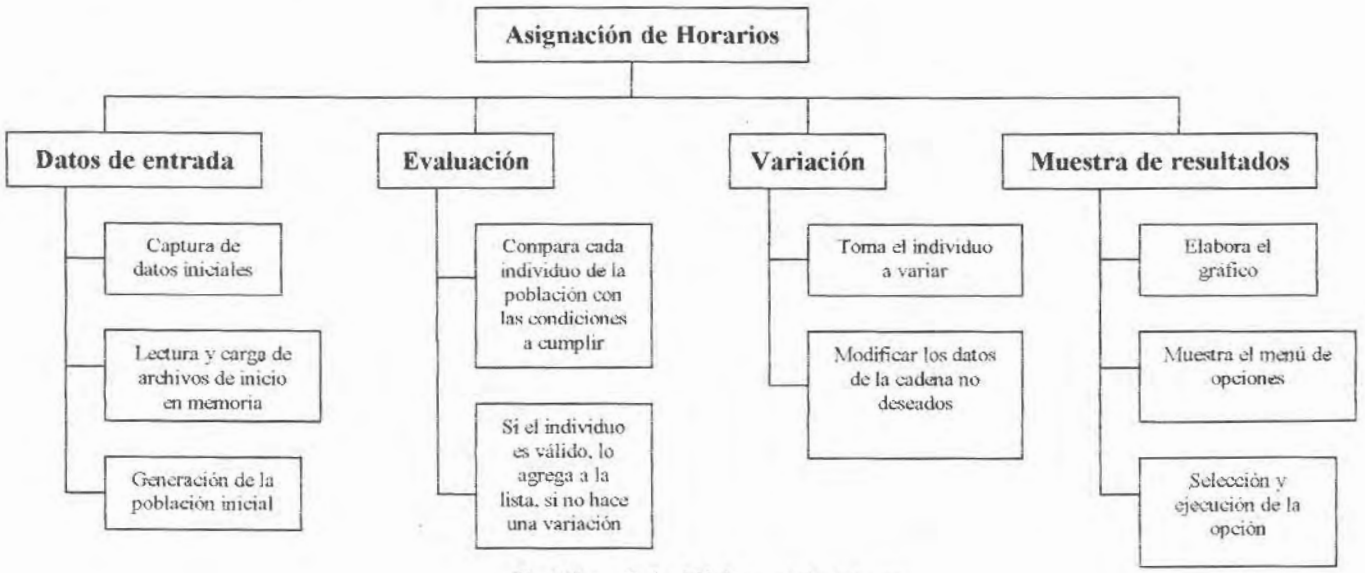
4.3. Descripción del algoritmo implementado

El programa para computadora que se propone, está compilado en lenguaje de programación Borland C y se utilizó la programación estructurada para la realización del programa.

Se eligió este lenguaje y este tipo de programación por la facilidad con la que se pueden realizar y presentar las rutinas a utilizarse para la solución del problema de asignación de horarios, que hacen referencia a los algoritmos genéticos, además de ser un lenguaje de programación que no requiere gran cantidad de recursos computacionales para un buen funcionamiento de los programas que en él se realicen.

4.3.1. Módulos del programa

La estructura del código que compone este programa de asignación de horarios es la siguiente:



Gráfica 4.1. *Módulos del sistema*

4.3.1.1. Módulo de datos de entrada

El programa de asignación de horarios presenta cuatro módulos principales, el módulo referente a **Datos de Entrada** se encarga de la captura de los siguientes datos iniciales:

- **Período para el cual se realiza la asignación de horarios.**

Existen dos periodos, cada uno tiene una duración de seis meses, el primer periodo abarca del mes de septiembre a febrero, y el segundo del mes de marzo a agosto. Este dato se solicita al usuario, él sólo tecleará el valor 1 o 2 según sea su elección, con este dato el sistema sabrá que semestres y materias se deberán considerar para la distribución en cada grupo.

El primer período incluye los semestres de primero, tercero y quinto. Para el segundo periodo se tienen los semestre segundo, cuarto y sexto, en ningún caso se ha dado una mezcla de estos periodos.

- **Tamaño de la población.**

Se solicita al usuario que teclee el número de individuos que formarán la población. Donde cada individuo representa una posible asignación de una materia en un determinado grupo. La cantidad de individuos que pueden formar a la población dependerá de la capacidad de memoria de la maquina con la que se ejecute este programa, puede ser desde uno hasta más de mil individuos. El sistema que se propone, se ejecutó en un equipo con memoria RAM de 20Mb y un procesador 486, variando el tamaño de la población de 1 hasta 200 individuos como máximo.

- **Lectura y carga de archivos de inicio a memoria RAM de la computadora.**

Este proceso se encarga de abrir los archivos de datos necesarios para realizar el procedimiento. Se necesitan dos archivos de datos: archivo de materias y archivo de semestres. A continuación se describe cada uno de ellos.

1. **Archivo de materias.**

En este archivo se encuentran registradas cada materia con sus respectivos datos. Este archivo contiene números que indican cada caracteristica o dato de la materia que representan. El contenido de este archivo se muestra a continuación, con un encabezado que indica a que se refiere cada dato de la columna (ver tabla 4.1).

No. de materia	Semestre	Hrs. Teoría	Hrs. Práctica	Carrera	Nombre
1	1	5	0	0	matematics
2	1	3	0	0	Computac
3	1	4	0	0	Comunicaci
4	1	3	0	0	Inglés
5	1	4	0	0	Valores
14	3	5	0	0	Mat_tec
15	3	3	0	0	Historia
16	3	3	0	0	Com_cietec
17	3	3	0	0	Mec y calor
18	3	3	0	0	Cal_total
9	1	2	0	2	Org_de_Emp
10	1	11	0	2	Proc_cont
22	3	5	0	2	Oper_conta
23	3	5	0	2	Regl_tribu
24	3	4	0	2	Gest_merc
6	1	4	0	3	Org_de_empr
7	1	5	0	3	Proc_admti
8	1	4	0	3	Admon_vent
19	3	5	0	3	Admon_alma
20	3	5	0	3	Admon_produ
21	3	4	0	3	Ctrl_activ
34	5	11	0	2	Reexpresio
35	5	15	0	2	Planeación
29	5	3	0	2	Metodologia
30	5	3	0	0	ESEM
31	5	10	0	3	Ctlr_recur
32	5	10	0	3	Admon_suel
33	5	6	0	3	Ctrl_prest
37	5	5	13	1	Redes
36	5	2	6	1	Manej_sist
11	1	2	3	1	Estruct_com
12	1	1	3	1	Oper_sist
13	1	1	3	1	Amb_graf
27	3	1	3	1	Mat_discret
28	3	1	3	1	Programac
25	3	1	2	1	Inst_amb
26	3	1	2	1	Sist_comp

Tabla 4.1. *Almacena los datos relacionados a las materias.*

En esta tabla se muestran las materias que corresponden al período septiembre-febrero, el registro en este archivo, se hace de manera directa, es decir se abre el archivo como tipo texto, para esto se puede utilizar el comando edit de MS-DOS o el bloc de notas de Windows. El registro de materias es lineal, es decir, cada línea del archivo representa los datos de una materia, los datos mostrados en la tabla son una parte del total de materias, en el momento que se implante en el plantel CONALEP, se tendrá el registro completo de todas las materias de los dos períodos.

La primer columna indica el número consecutivo de materia en la lista que contiene el personal administrativo, como se observa, este número se encuentra en desorden, pues

las materias han sido colocadas según el siguiente criterio, consiste en colocar las materias con horas teóricas en primer lugar y al final las materias que tienen horas prácticas, el número que se encuentra en esta columna es asignado según el semestre al que corresponde cada materia (de menor a mayor).

En la segunda columna se encontrará el valor que indica el semestre al que corresponde cada materia, de uno a seis, de primero a sexto respectivamente.

La tercera columna presenta las horas teóricas de cada materia, estas horas indican que se deben asignar las clases correspondientes en un salón en el que no es necesario se tenga una computadora.

La siguiente columna indica las horas prácticas de cada materia, esto quiere decir que las horas clase de este tipo se deberán asignar en un laboratorio de cómputo, ya que requiere del uso de una computadora.

El dato correspondiente a la carrera a la que se debe asignar cada materia se encuentra en la penúltima columna, en esta columna se encontrará una de cuatro opciones:

0 -> Si la materia se debe asignar a todos los grupos que correspondan con el semestre que indica, sin importar la carrera a la que pertenezcan.

1 -> Si la materia se debe asignar a los grupos que correspondan con el semestre que indica y la carrera de informática.

2 -> Si la materia se debe asignar a los grupos que correspondan con el semestre que indica y la carrera de contabilidad.

3 -> Si la materia se debe asignar a los grupos que correspondan con el semestre que indica y la carrera de administración.

Por último se encontrará el nombre de la materia, este dato será útil al momento de presentar los resultados en pantalla para su consulta o su modificación. De esta manera se tendrá conocimiento de que materia se está asignando, así como cual se está modificando.

2. Archivo de semestres.

El archivo correspondiente a semestres lleva el siguiente registro (ver tabla 4.2):

Semestre	Grupos	Turno	Carrera
1	4	0	1
1	3	0	2
1	1	0	3
2	0	0	1
2	0	0	2
2	0	0	3
3	1	0	1
3	0	0	2
3	0	0	3
3	2	1	1
3	2	1	2
3	1	1	3
4	0	1	1
4	0	1	2
4	0	1	3
5	1	0	1
5	2	1	1
5	2	1	2
5	1	1	3
6	0	1	1
6	0	1	2
6	0	1	3

Tabla 4.2. *Contenido referente a los semestres*

La descripción de esta tabla es la siguiente:

Este archivo registra cada semestre, el total de grupos que se tienen para este semestre, así como el turno que se les ha asignado, también se encuentra la carrera a la que pertenece cada semestre que se indica, al igual que el registro de materias, éste es por línea, en cada línea del archivo se encuentran datos que corresponden sólo al semestre que se indica, ningún registro se repite.

Cabe aclarar que el registro que se lleva en ambos archivos es de números y no de cadenas de caracteres, esto ahorra memoria al momento de cargar los archivos y el manejo de los datos se simplifica, pues es más sencillo manejar un sólo número que toda una cadena de caracteres. Al presentar los resultados, se convierten o se relacionan dichos números con su respectiva cadena, que puede indicar el grupo o la materia a la que corresponde.

En la primer columna se encuentra el dato que indica el semestre, este dato es un valor de uno a seis indicando desde el primer semestre a sexto respectivamente.

A continuación se tendrá el número de grupos que se han formado para este semestre, por infraestructura y demanda nunca se ha tenido más de cinco grupos en una carrera del mismo semestre.

La siguiente columna representa el turno que se ha asignado a estos grupo, este puede ser un valor de 0 para el turno matutino o 1 para el turno vespertino.

La última columna registra el dato que corresponde a la carrera para la cual se han asignado estos grupos y este semestre. En esta tabla para representar la carrera se tienen solo tres opciones:

- Informática
- Contabilidad
- Administración.

Una vez que se han leído y cargado en memoria estos archivos, se forman dos listas simples. Para poder guardar los datos tanto de materias como de semestres en la memoria de computadora se requiere la declaración de estructuras que contendrán los datos que se toman del archivo correspondiente. A continuación se explican dichas estructuras de datos.

4.3.1.1.1. Declaración de estructuras

La instrucción *struct* y el tipo de datos *unsigned* han sido de gran utilidad para el funcionamiento general del programa y en particular para el almacenamiento y manejo de estas estructuras.

La declaración de las estructuras a utilizar es de la siguiente forma:

- Estructura correspondiente a la tabla **Materias.**

```
struct materias{
unsigned materia : 6;
unsigned semestre : 3;
unsigned hrs_teo : 4;
unsigned hrs_prac:4;
unsigned carrera:2;
char nom[25];
struct materias *sig;
};
```

El número de bits requerido dependerá de la cifra a almacenar en el campo correspondiente[22].

- Estructura para la tabla **semestres o totales.**

```
struct totales{
unsigned semestre : 3;
unsigned grupos : 4;
unsigned turno : 1;
unsigned carrera: 2;
unsigned hrs_tot: 10;
unsigned hrs_grupo:10;
unsigned mprac:4;
unsigned mteo:4;
struct totales *sig;
};
```

Esta declaración abarca los campos descritos en la tabla 4.2 correspondiente a semestres, y se agregan otros campos para mejorar la utilización de los datos que se almacenan en esta estructura.

En ambas listas, al hacer un registro, se forma un nodo o un espacio en memoria destinado a guardar los datos que correspondan con cada estructura, estos nodos se unen con un apuntador de estructura, este se declara dentro de cada estructura de la forma siguiente: *struct totales *sig;* Además de unir los nodos de la lista, este apuntador permite pasar de un nodo a otro.

En forma de esquema, estas listas se representan a continuación:

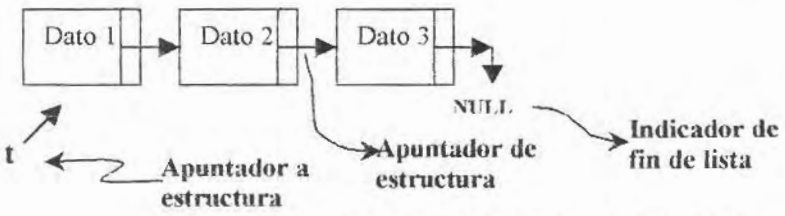


Figura 4.1. *Representación de una lista ligada*

En páginas posteriores se muestra el algoritmo para este procedimiento.

4.3.1.2. Generación de la población inicial

Parte de los datos de entrada es la generación de la población inicial, esta población se genera tomando como inicio el valor que el usuario tecleo como respuesta a cuántos individuos tendrá la población.

Cada individuo que se genere estará representado por la siguiente estructura:

```
struct asignaciones{
unsigned materia : 6;
unsigned semestre : 3;
unsigned turno : 1;
unsigned dia : 3;
unsigned hora : 5;
unsigned grupo : 3;
unsigned carrera:2;
unsigned lab:2;
unsigned hrs_tot:5;
unsigned hrs_prac:5;
char nom[25];
unsigned fijar : 8;
struct asignaciones *sig;};
```

Los campos que se presentan en esta estructura, son las características que cada individuo posee, estas serán útiles cuando se comparen con las condiciones que el programa y el mismo procedimiento exige para ser una asignación de materias válida y apta. Cada individuo es único, no deberá haber individuos que coincidan en la totalidad de sus características.

Una vez que se ha generado la población inicial y se han cargado los datos necesarios en memoria, el programa está listo para comenzar la siguiente fase del

procedimiento, esta fase se refiere a la aplicación de los algoritmos genéticos a los individuos generados en la población inicial, esto se explica a continuación.

4.3.1.3. Módulo de Evaluación

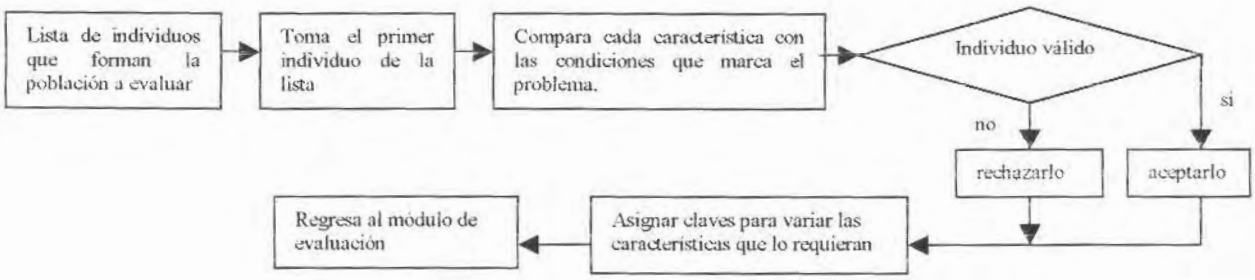
El módulo de ***Evaluación*** se encarga de realizar los siguientes procesos:

- Compara cada individuo de la población con las condiciones que debe cumplir para ser un individuo aceptado. Se dice que es un individuo aceptado cuando cada característica que almacena en su información es igual a las condiciones que en ese momento exige el programa.

- Si el individuo a cumplido satisfactoriamente las condiciones exigidas, el individuo se agrega a la lista que almacena a todos los individuos hasta ese momento aptos. Para que el individuo pueda formar parte de la lista, se revisa de principio a fin, con el objetivo de no repetir características de los individuos, en el caso de ya existir en la lista dicho individuo, se rechaza y se le impone una clave que indica que es un individuo a variar en sus características, en caso contrario, se anexa al final de la lista. Al agregar el individuo, se copian las características de una lista a otra, y se asigna una clave para variar las características que lo integran. Cabe mencionar que el programa empieza asignando horarios a las materias que tienen preferencias, para después continuar con el resto de materias.

  Para el caso en el que alguna de las características del individuo no son aceptables, se asigna una clave que indica que características se deben variar por no ser aceptables y no modificar las que si son aceptables.

Una representación gráfica del funcionamiento de este proceso es la siguiente:



Gráfica 4.2. *Módulo de evaluación de la población*

4.3.1.4. Módulo de Variación

Una vez que se ha evaluado la población de principio a fin y no se ha alcanzado el objetivo del programa, el siguiente proceso es variar los individuos, para obtener nuevos individuos y que ofrezcan nuevas posibilidades de formar individuos aptos, este proceso lo

realiza el módulo de ***Variación***, este módulo que en un principio parece diferente, juega el mismo papel que la cruza y la mutación, ya que a partir de los cromosomas existentes crea cromosomas o hijos cada vez más aptos, los cuales permiten obtener la solución deseada , donde parte del material genético permanece (el que es apto) y la otra parte de su material genético es cambiada (el cual tiene una marca para ser cambiada). Esta rutina aunque con nombre diferente realiza ambas funciones de cruza y mutación, permitiendo con ello que el proceso para encontrar la solución apropiada sea más rápido.

Su funcionamiento es de la siguiente forma:

- Desde el primer individuo al último de la lista revisa la clave que se le ha asignado en el módulo de evaluación. El proceso de *variación* determina que característica deberá variar en cada individuo, que puede ser una o todas las características que lo componen, esto dependerá de la clave que se le ha asignado para ser considerado como un individuo apto.

- La variación de un individuo consiste en asignar un nuevo valor aleatorio a cada característica a modificar, este nuevo valor depende de un rango y a su vez depende de la longitud en bits de cada campo.

- Al terminar las variaciones requeridas en la población, regresa al módulo de evaluación, este ciclo continúa hasta alcanzar el objetivo deseado.

Este proceso se ilustra en la siguiente figura 4.3.

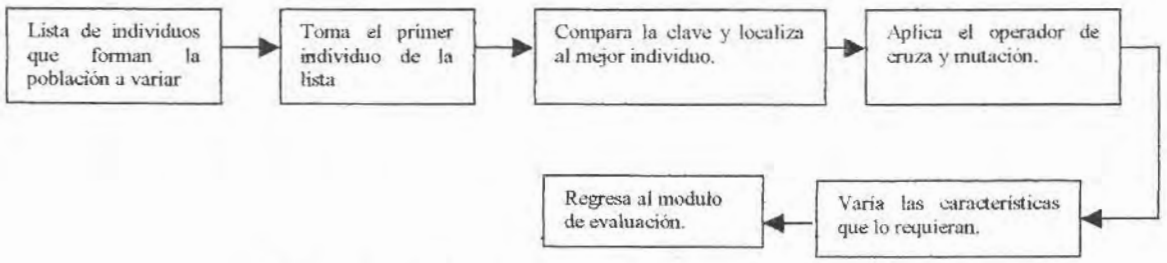


Figura 4.3. *Funcionamiento del módulo de variación.*

4.3.1.5. Módulo de muestra de resultados

El módulo de **muestra de resultados** se encarga de realizar una presentación gráfica en pantalla, presentando una tabla que permite visualizar el horario de cada grupo o laboratorio de computo, así como la presentación de un menú para consulta de los resultados obtenidos en el programa.

4.4. Algoritmos

Un **algoritmo** es una serie de pasos que indican el orden que lleva un determinado proceso para su realización.

A continuación presentamos las subrutinas que componen al algoritmo que realiza la asignación de horarios aplicando la técnica de algoritmos genéticos. Se omiten los módulos de muestra de resultados, ya que aparte de estar formados por una gran lista de instrucciones, no es la parte importante del algoritmo realizado.

4.4.1. Algoritmo para la función principal

El algoritmo para el bloque principal hace un llamado a las funciones de captura y de lectura de datos, a la función de generación de la población inicial, estos procesos serán la base para el proceso de asignación de horarios, la siguiente parte de la función principal es un ciclo que evalúa y varía a los individuos de la población hasta alcanzar el número de asignaciones requeridas y que están determinadas por el número de grupos que componen a cada semestre, el número de materias y el número de horas que corresponden a cada una de ellas. Como se ilustra a continuación.

1 Limpiar pantalla
2 Mostrar mensaje *"Pulse el valor que corresponda al período de la actual distribución"*.
3 Mostrar en pantalla las siguientes opciones:
    1. Período Septiembre-Febrero.
    2. Período Marzo-Agosto.
4 Esperar respuesta.
5 Almacenar valor de respuesta en *opc*.
6 Mostrar mensaje *"Tamaño de la población a generar"*.
7 Esperar respuesta.
8 Almacenar respuesta en *pob*.
9 *Lee y carga* en memoria los archivos de materias y semestres
10 *Genera* la población inicial
11 *Desde* el primer registro de la lista de semestres *hasta* el último
    a. *Mientras* el campo grupos de la lista de semestres sea *mayor de* cero
        i. *Mientras* el total de individuos asignados sea *menor que* el total de horas a asignar para cada grupo
            1. *Ejecuta* el proceso evaluación de la población
            2. *Ejecuta* el proceso variación de la población
    b. *Avanza* al siguiente registro
12 *Termina*

Su diagrama de flujo es el siguiente:

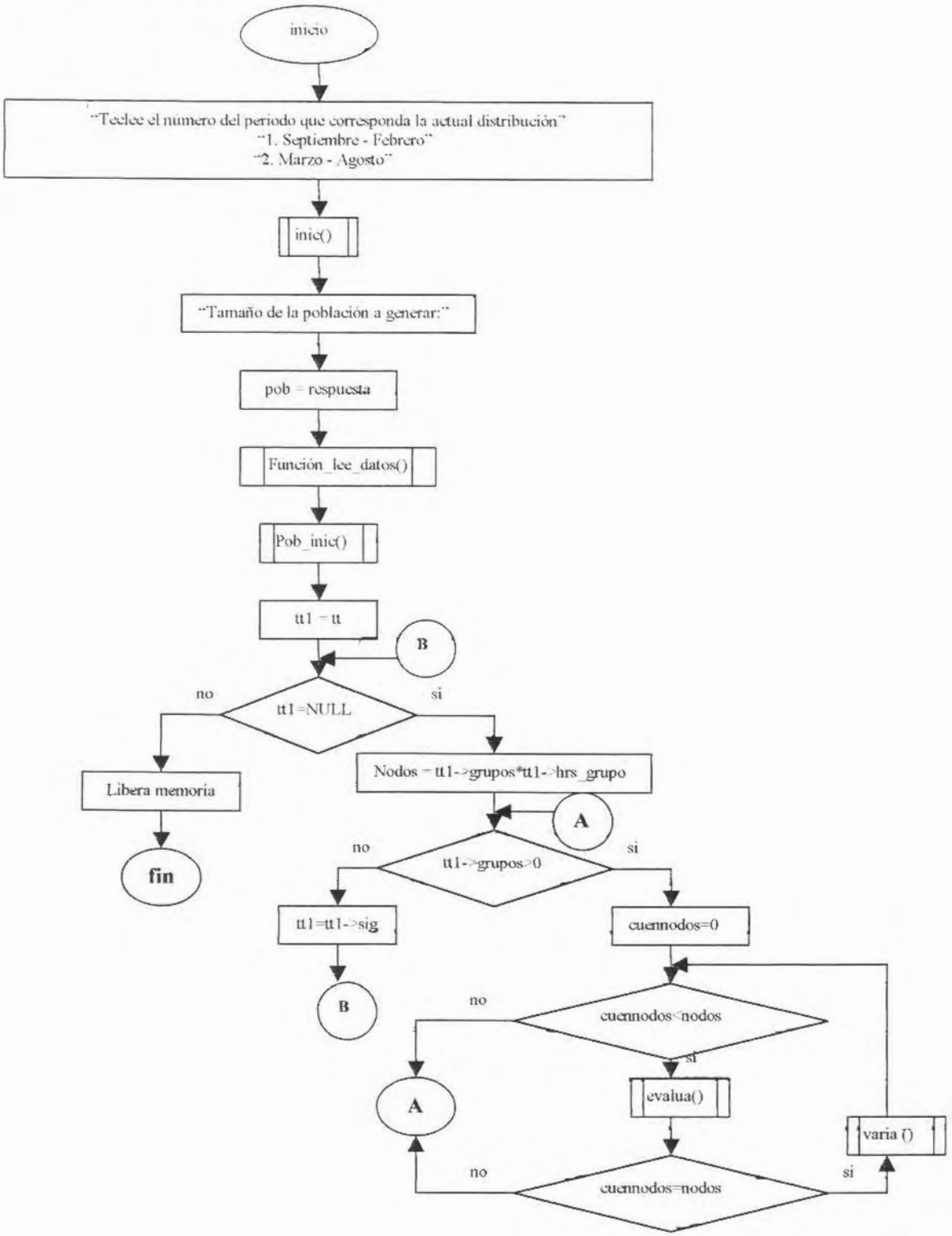


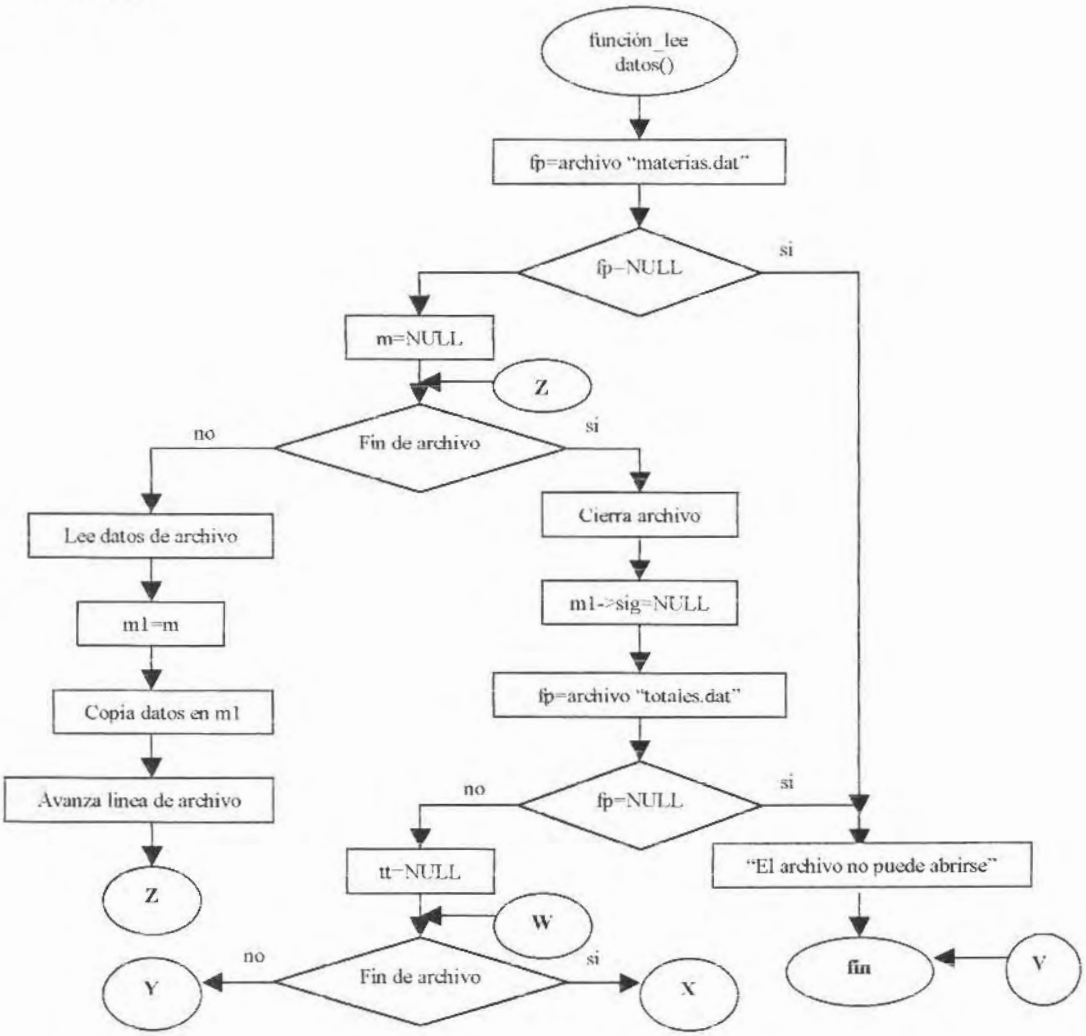
Diagrama de flujo 4.1. Programa principal

4.4.2. Algoritmo de lectura y carga de archivos de materias y semestres

En esta rutina el programa lleva acabo las siguientes instrucciones:

1. Abrir archivo correspondiente a materias.
2. *Mientras* no sea el final del archivo
   2.1 Crear nodo en memoria
   2.2 Leer línea y almacenar en memoria
   2.3 Regresa al paso 2.
3. Abrir archivo correspondiente a semestres
4. *Mientras* no sea el final del archivo
   4.1 Crear nodo en memoria
   4.2 Leer línea y almacenar en memoria
   4.3 Regresa al paso 4.

Diagrama de flujo:



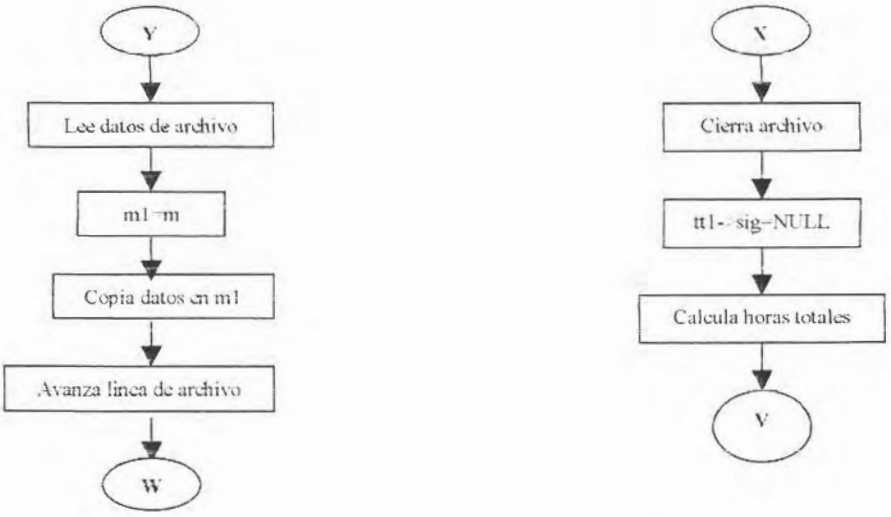


Diagrama de flujo 4.2. *Lectura y carga de archivos*

4.4.3. Algoritmo que genera la población inicial

En esta rutina se genera la población inicial con el número total de individuos dado por el usuario, realizando las siguientes instrucciones:

1. *Desde* i=1 *hasta* número de individuos de la población
   - 1.1 Crea nodo en memoria
   - 1.2 *Si* es primer nodo
     - 1.2.1 Asigna puntero de inicio de lista (*as)
   - 1.3 Asigna puntero de recorrido de lista al nodo creado (*as1)
   - 1.4 Asigna a cada campo del nodo creado los siguientes valores
     - 1.4.1 as1->materia=1+random(37);
     - 1.4.2 as1->semestre=1+random(6);
     - 1.4.3 as1->turno=random(2);
     - 1.4.4 as1->dia=1+random(5);
     - 1.4.5 as1->hora=7+random(15);
     - 1.4.6 as1->grupo=1+random(8);
     - 1.4.7 as1->carrera=1+random(3);
     - 1.4.8 as1->lab=0;
     - 1.4.9 as1->hrs_tot=0;
     - 1.4.10 as1->fijar=0;
   - 1.5 incrementa i
2. Asigna NULL al último nodo como indicador de fin de lista
3. Regresa.

En este algoritmo se muestran algunas líneas en forma de código de la manera en que se escribe en lenguaje C, la variable *as1* representa un apuntador a la estructura y a su vez nos permite tener acceso a cada campo de la estructura, el valor NULL para un apuntador es como asignar el valor cero a una variable de tipo entero. La asignación de valores a cada campo es aleatoria (por medio de la función *random*), lo que permite tener

varios puntos de comparación iniciales, y enfocarse al valor que se acerca más a la solución buscada, su diagrama de flujo es el siguiente:

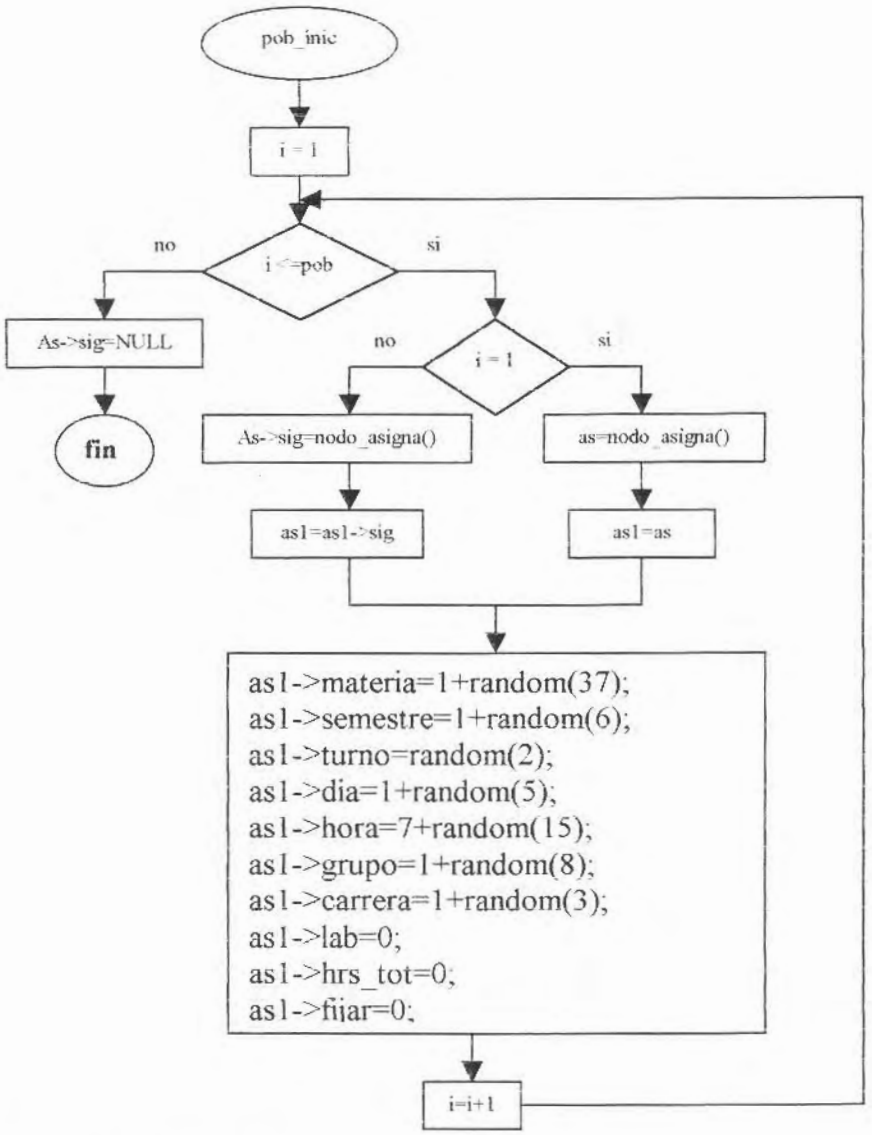


Diagrama de flujo 4.3. *Generación de la población inicial*

4.4.4. Algoritmo de Evaluación de la población

Esta rutina se encarga de verificar que los individuos cumplan con las especificaciones requeridas

1. *Desde* el primer individuo *hasta* el último de la población a evaluar
   1.1 *Si* campo semestre es igual a semestre a buscar
      1.1.1 *Desde* la primer materia *hasta* la última de la lista de materias
         1.1.1.1 *Si* campo materia es diferente a materia a buscar
            1.1.1.1.1 Avanza a la siguiente materia de la lista
         1.1.1.2 *En otro caso* terminar búsqueda
      1.1.2 *Si* la materia buscada existe
         1.1.2.1 *Si* el campo hrs_ prac es diferente de 0
            1.1.2.1.1 *Si* el campo materias prácticas es mayor que las materias prácticas asignadas
               1.1.2.1.1.1 Ejecuta el proceso f_evalua que continua la evaluación
            1.1.2.1.2 En otro caso
               1.1.2.1.2.1 *Si* las materias prácticas asignadas son igual al campo materias prácticas
                  1.1.2.1.2.1.1 Ejecuta el proceso f_evalua
      1.1.3 *En otro caso* rechazar individuo
   1.2 *En otro caso* rechazar individuo
2. regresar

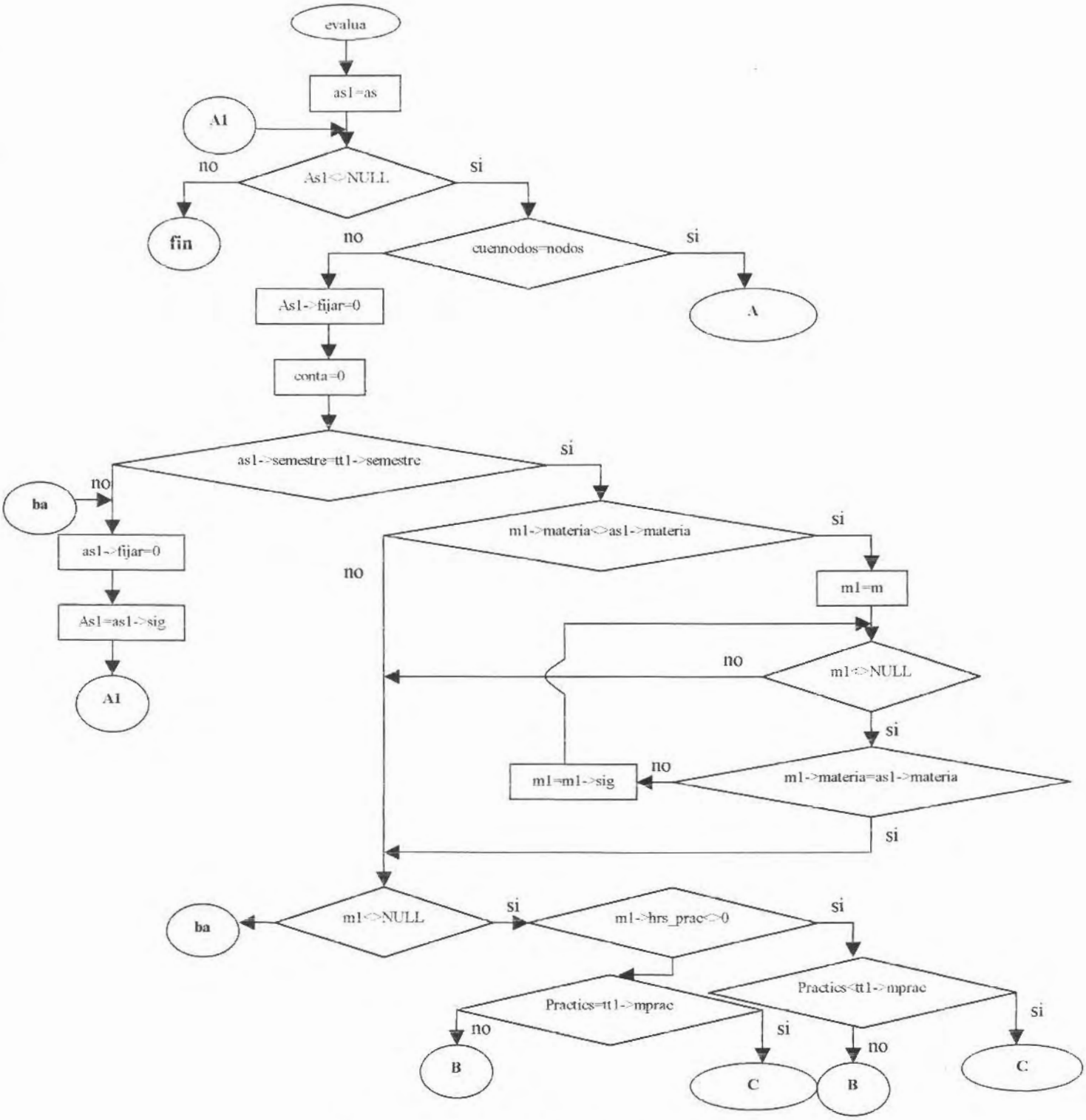


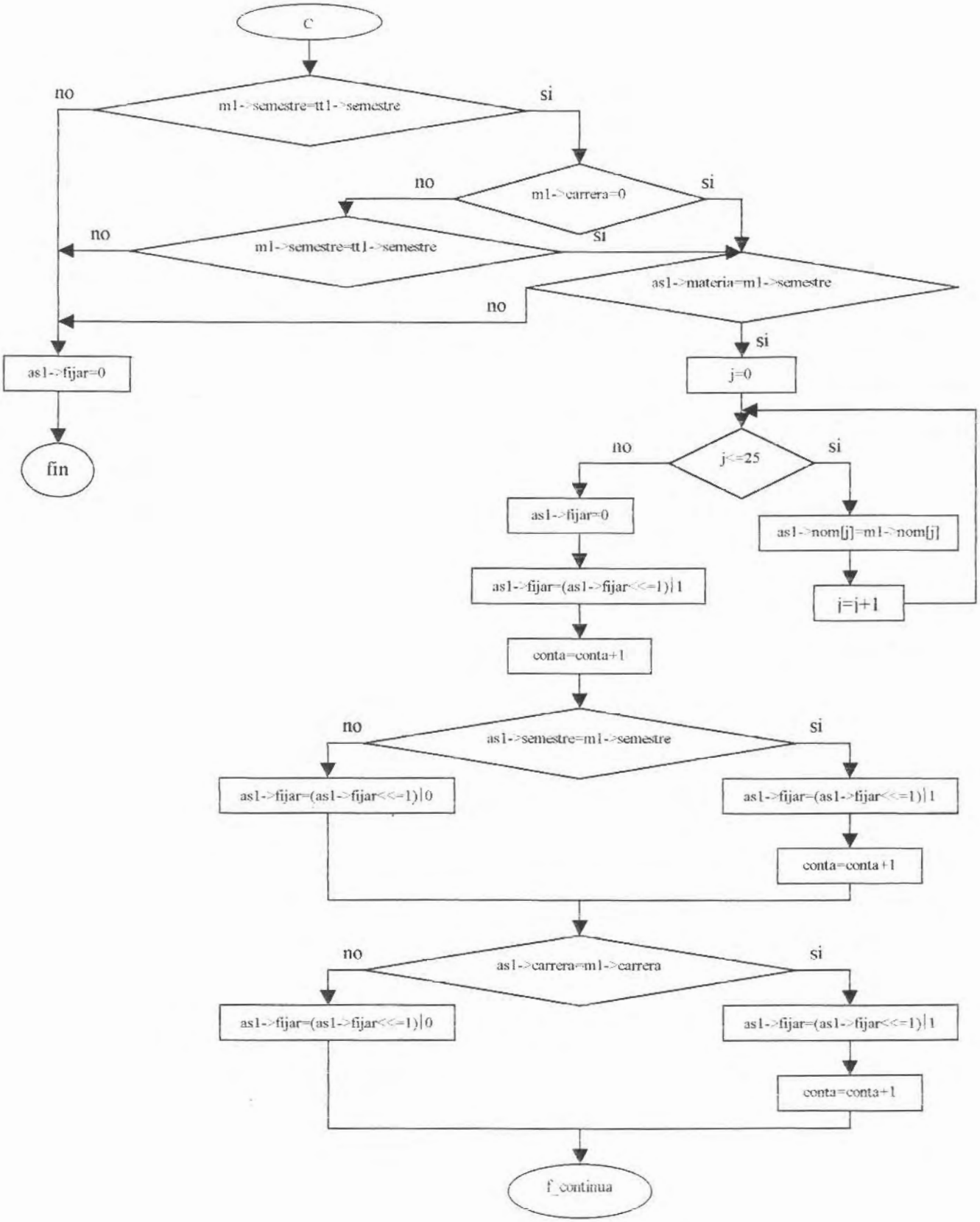
Diagrama de flujo 4.4 *Evaluación de la población*

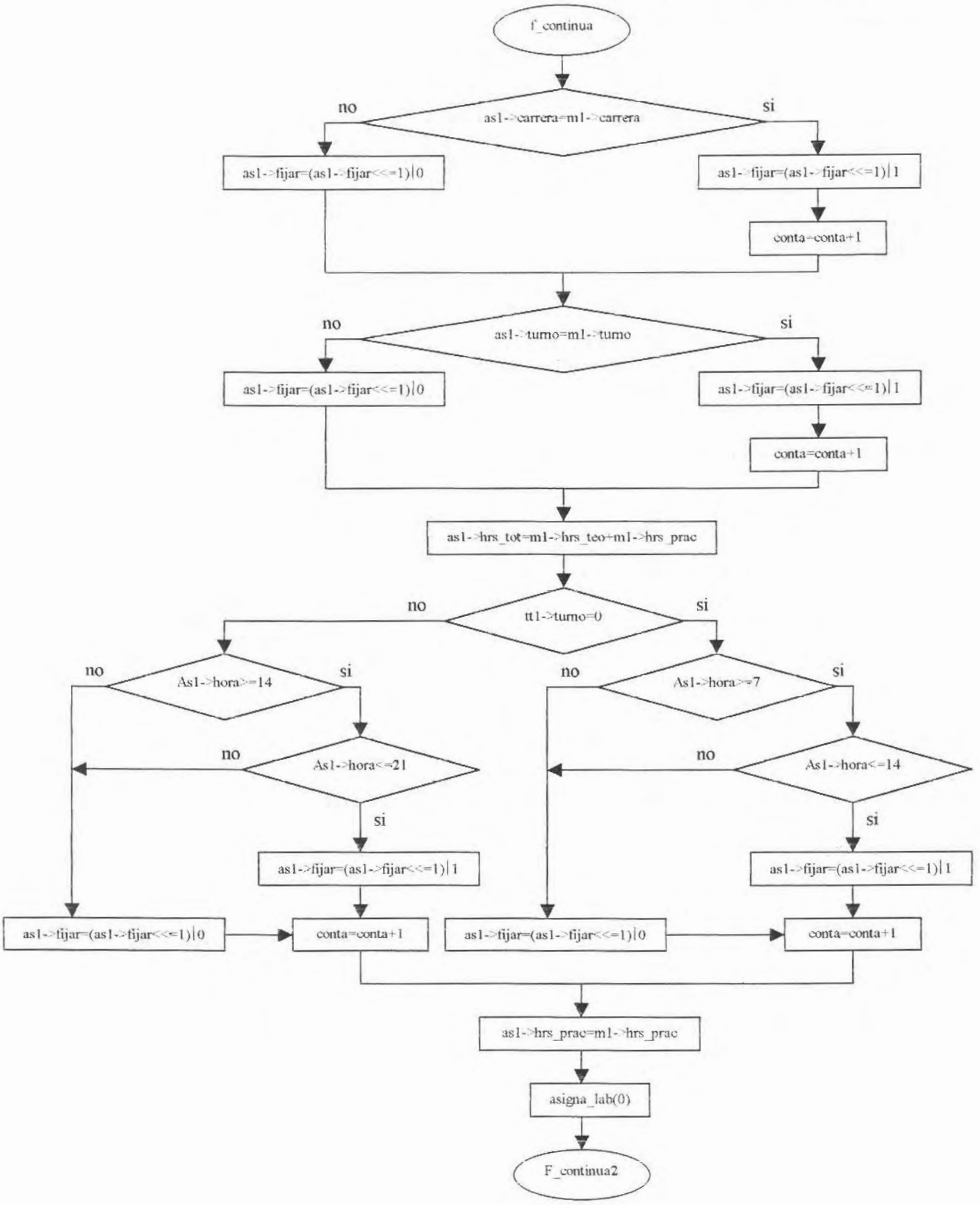
4.4.4.1. Algoritmo para el proceso f_evalua

Esta función es el complemento de la función de evaluación antes descrita, se encarga de tomar el individuo que ha cumplido con las restricciones del problema y lo envía a una lista de individuos que guarda a los que ya han sido aceptados.

1. *Si* el campo semestre de la lista de materias es igual al campo semestre de la lista de semestres
   1.1 *Si* el campo carrera de materias es igual a cero ó es igual al campo carrera de la lista de semestres
      1.1.1 Copia el nombre de la materia al campo nombre del individuo que se esta evaluando
      1.1.2 Asigna clave para no modificar este dato
      1.1.3 *Si* el campo semestre del individuo es igual al campo semestre de la materia seleccionada
         1.1.3.1 Asigna clave para no modificar este dato
      1.1.4 *En otro caso* asignar clave para modificar este dato
      1.1.5 *Si* el campo carrera del individuo es igual al campo carrera de la lista de semestres seleccionado
         1.1.5.1 Asigna clave para no modificar este dato
      1.1.6 *En otro caso* asignar clave para modificar este dato
      1.1.7 *Si* el campo turno del individuo es igual al campo turno de la lista de semestres seleccionado
         1.1.7.1 Asigna clave para no modificar este dato
      1.1.8 *En otro caso* asignar clave para modificar este dato
      1.1.9 *Si* el campo turno de la lista de semestres seleccionado es igual a cero
         1.1.9.1 Si el campo horas del individuo es mayor o igual a 7 y menor o igual a 14
            1.1.9.1.1 Asigna clave para no modificar este dato
         1.1.9.2 En otro caso asignar clave para modificar este dato
      1.1.10 *En otro caso*
         1.1.10.1 *Si* el campo horas del individuo es mayor o igual a 14 y menor o igual a 20
            1.1.10.1.1 Asigna clave para no modificar este dato
         1.1.10.2 *En otro caso* asignar clave para modificar este dato
      1.1.11 *Si* el campo día del individuo es mayor o igual a 1 y menor o igual a 5
         1.1.11.1 Asigna clave para no modificar este dato
      1.1.12 *En otro caso* asignar clave para modificar este dato
      1.1.13 *Si* el número de características a no modificar es igual con 6
         1.1.13.1 Ejecuta proceso lista
   2.1 *En otro caso* asigna clave para modificar todas las características del individuo
3. *En otro caso* asigna clave para modificar todas las características del individuo.
4. Regresa

Su diagrama de flujo es el siguiente:




f_continua
as1->carrera=m1->carrera
no
si
as1->fijar=(as1->fijar<<=1)|0
as1->fijar=(as1->fijar<<=1)|1
conta=conta+1
as1->turno=m1->turno
no
si
as1->fijar=(as1->fijar<<=1)|0
as1->fijar=(as1->fijar<<=1)|1
conta=conta+1
as1->hrs_tot=m1->hrs_teo+m1->hrs_prac
tt1->turno=0
no
si
As1->hora>=14
no
si
As1->hora<=21
si
as1->fijar=(as1->fijar<<=1)|1
as1->fijar=(as1->fijar<<=1)|0
conta=conta+1
As1->hora>=7
no
si
As1->hora<=14
si
as1->fijar=(as1->fijar<<=1)|1
as1->fijar=(as1->fijar<<=1)|0
conta=conta+1
as1->hrs_prac=m1->hrs_prac
asigna_lab(0)
F_continua2

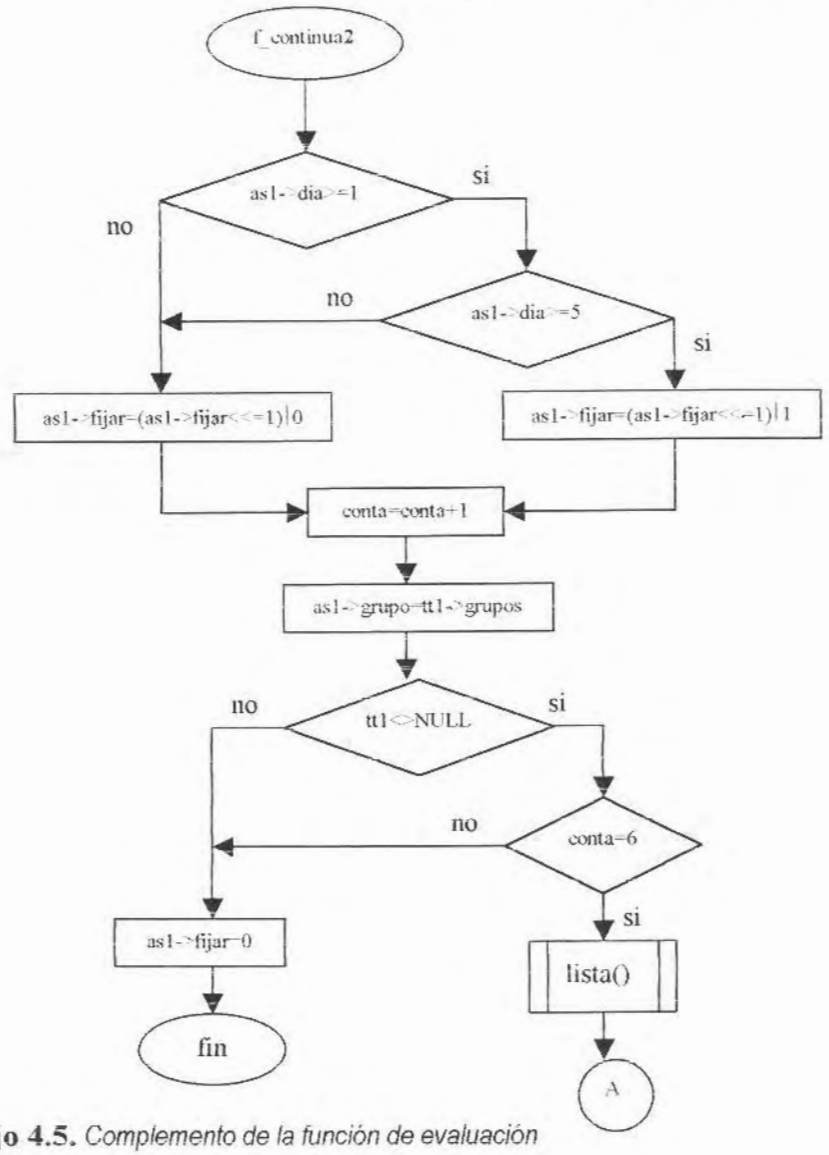


Diagrama de flujo 4.5. *Complemento de la función de evaluación*

4.4.5. Algoritmo para el proceso lista

1. *Si* la lista de individuos esta vacía
   1.1 Crear espacio en memoria
   1.2 Asignar apuntador de inicio de lista (*as2)
   1.3 Copia datos del individuo valido al nuevo nodo creado
2. *En otro caso*
   2.1 *Desde* el primer individuo insertado *hasta* el último
      2.1.1 Comparar las características de cada individuo insertado con el nuevo individuo a insertar
      2.1.2 *Si* las características son iguales
         2.1.2.1 Rechazar individuo a insertar, asignando la clave para modificarse
         2.1.2.2 Termina la comparación
      2.1.3 En otro caso

2.1.3.1 Avanzar al siguiente individuo registrado

2.2 *Si* el individuo a insertar no existe en la lista

2.2.1 Crear espacio en memoria

2.2.2 Colocar al final de la lista el nuevo nodo creado

2.2.3 Copiar datos del individuo valido al nuevo nodo creado

3. Asignar clave al individuo valido para modificar sus características
4. Regresar

Su diagrama de flujo se muestra a continuación:

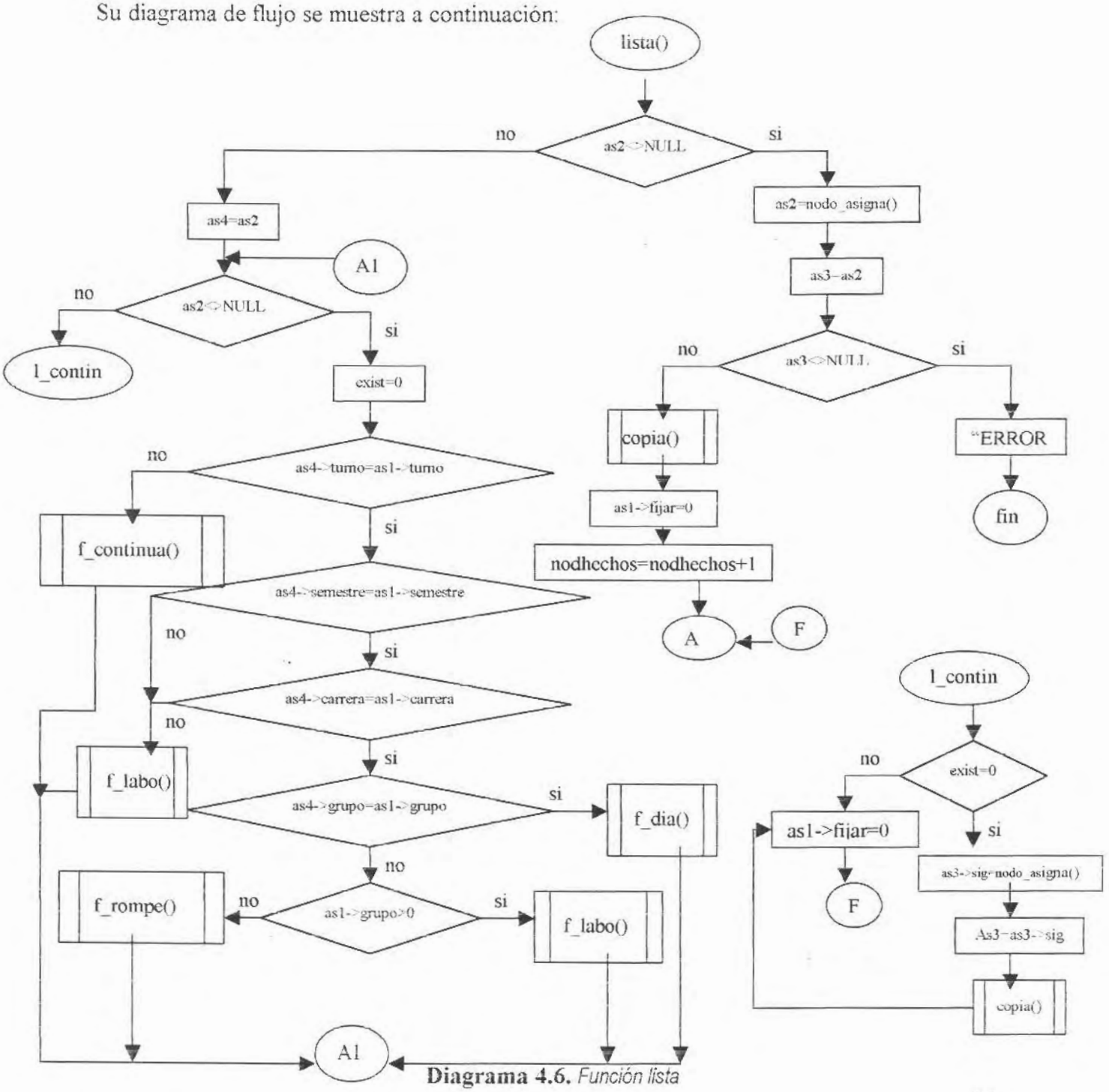
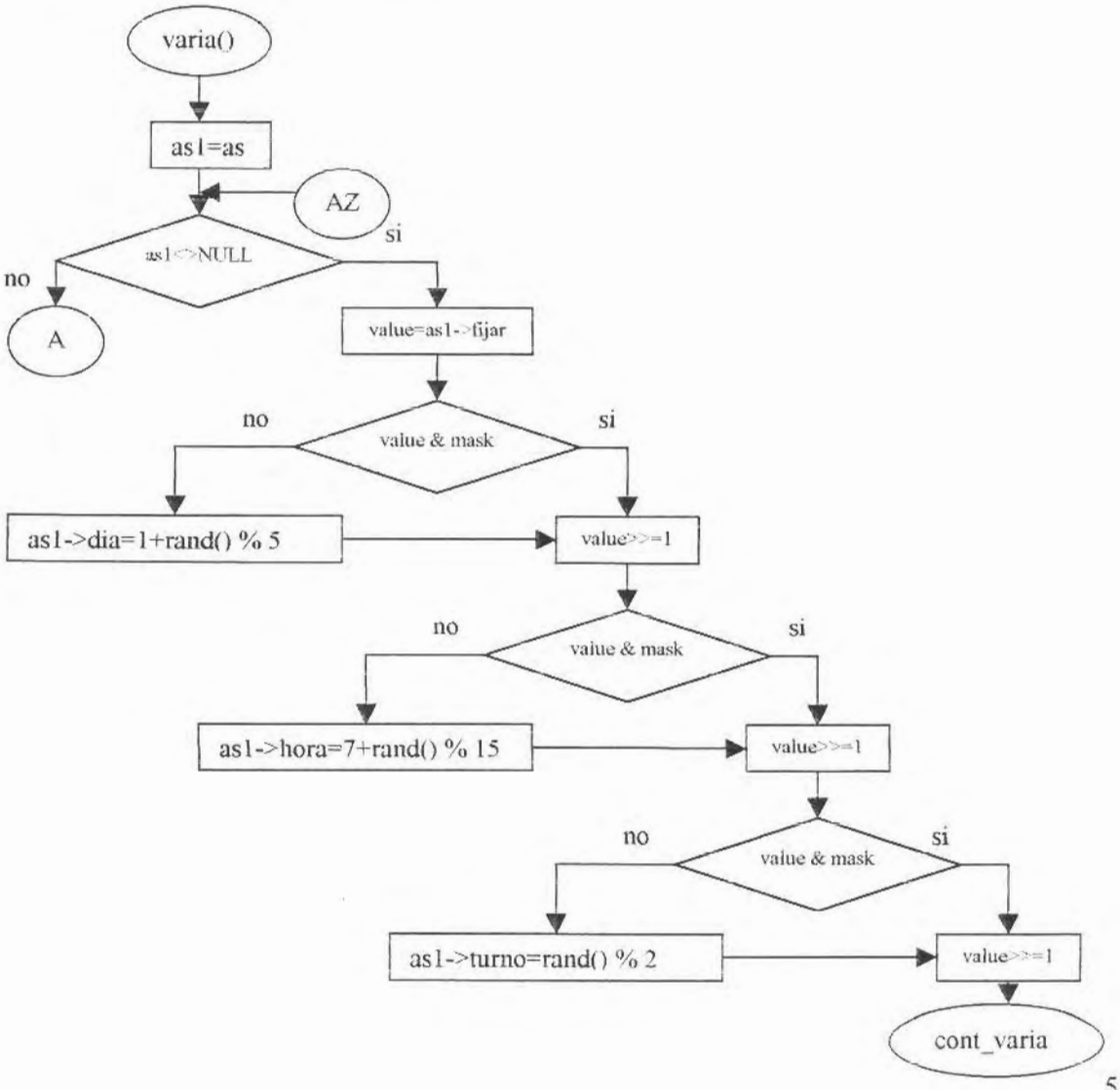


Diagrama 4.6. *Función lista*

4.4.6. Algoritmo de Variación de la población

1. *Desde* el primer individuo *hasta* el último de la población
   1.1 Comparar clave del individuo a modificar
   1.2 *Si* clave indica modificar campo día
      1.2.1 as1->dia=1+rand() %5
   1.3 *Si* clave indica modificar campo hora
      1.3.1 as1->hora=7+rand()%15
   1.2 *Si* clave indica modificar campo turno
      1.2.1 as1->turno=rand() %2
   1.3 *Si* clave indica modificar campo carrera
      1.3.1 as1->carrera=1+rand()%3
   1.2 *Si* clave indica modificar campo semestre
      1.2.1 as1->semestre=1+rand() %6
   1.3 *Si* clave indica modificar campo materia
      1.3.1 as1->materia=1+rand()%37
2. Regresar



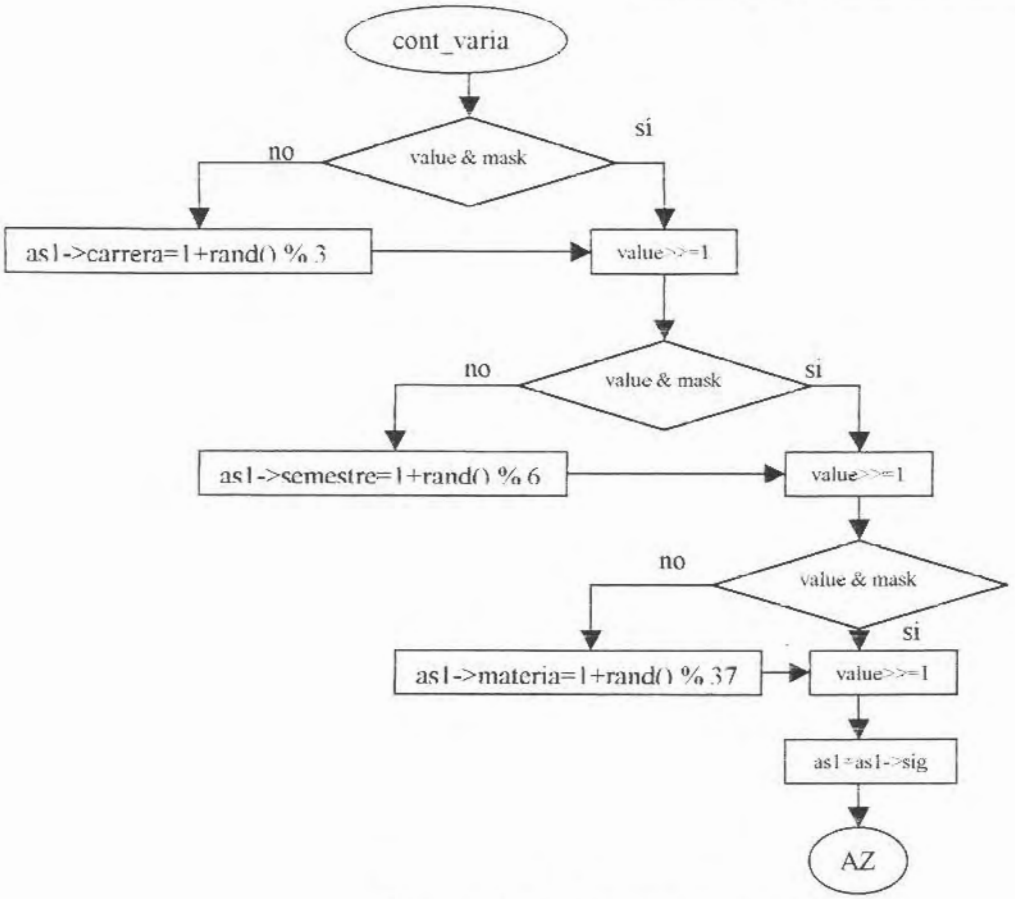
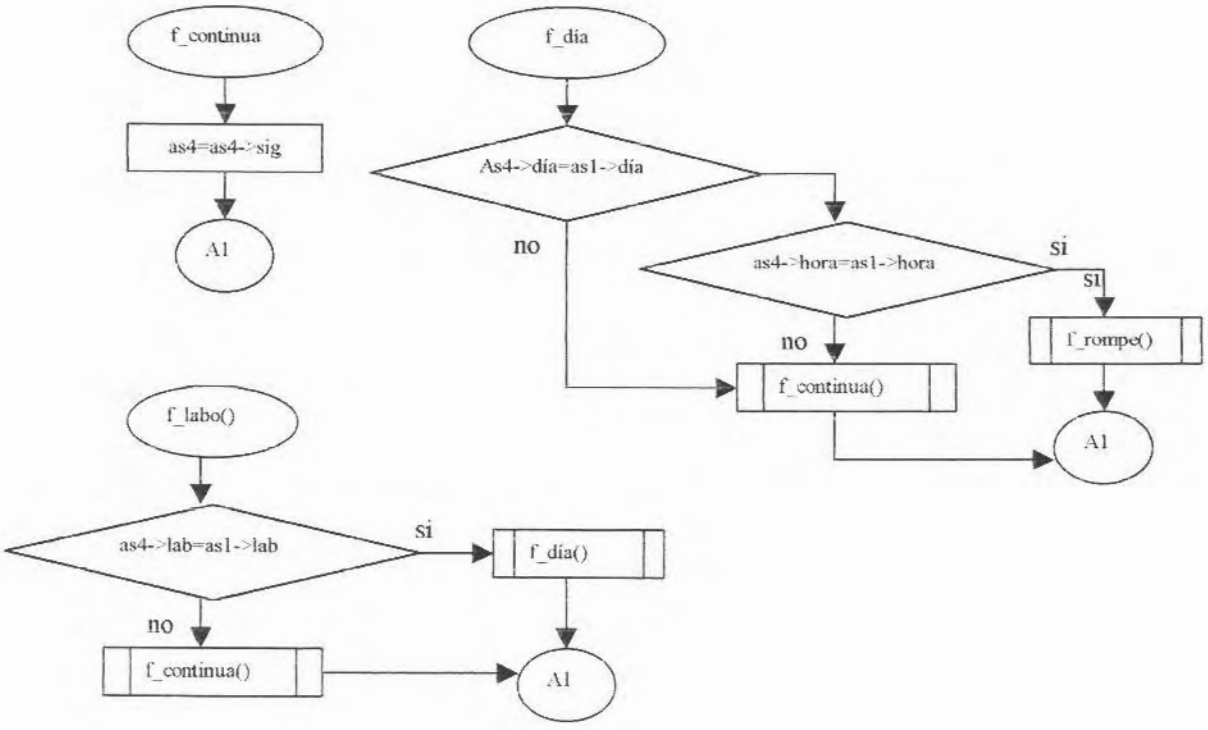
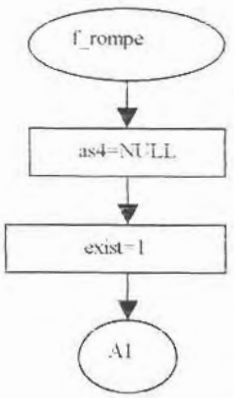
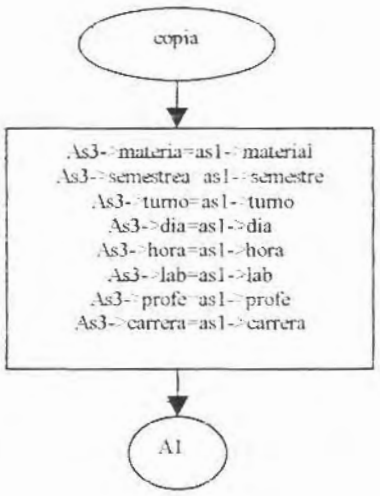


Diagrama 4.7. *Función variación*







4.5. Conclusiones

El algoritmo genético no es difícil de implementar, pues lo más complicado que realiza es copiar cadenas y variar cadenas parcialmente (operador de cruza y mutación).

El operador que hace la función de cruza es el operador variación, el cual modifica sólo aquellas características del cromosoma que lo requieren.

La función de evaluación que se utilizó, evalúa a los individuos de la población y aquellos que no cumplan las condiciones y restricciones del problema se les asigna un valor de cero por cada característica no válida, en caso contrario se les asigna el valor de uno.

Se utilizan estructuras con campos de bits en lugar de arreglos de valores enteros, esto ahorra espacio en memoria, aunque requiere de más líneas de código para el manejo de estas estructuras.

CAPÍTULO 5

PRUEBAS, RESULTADOS Y EJEMPLOS.

5.1. Introducción

Las pruebas que se realizan a un programa para computadora tienen el objetivo de encontrar errores en el funcionamiento general del sistema. Se estudia y analiza el rendimiento y funcionamiento de cada bloque del programa para después reunirlos y realizar la prueba al programa completo.

Las pruebas aplicadas a un programa se clasifican en dos tipos: *una prueba por parte de quien realizó el programa y una prueba por parte del cliente.*

La prueba por parte de quien realizó el sistema consiste en probar cada función y cada bloque del código que compone al programa. Deberá encontrar los errores y corregirlos ya sea modificando el código, agregando o quitando líneas de código.

La prueba por parte del cliente, principalmente se enfoca en la muestra de resultados, el cliente quiere ver en pantalla o en impresora lo que a él le va a facilitar su trabajo. El cliente busca facilidad y formas agradables de manejar sus datos, al consultar y modificar la información que se le muestra. El cliente no necesariamente debe saber como se programó la tarea que el requiere resolver e inclusive mientras menos datos tenga que proporcionar al programa es mejor para él.

5.2. Prueba al sistema por parte del programador

Al programa de asignación de horarios se aplicaron las siguientes pruebas. La *prueba por parte de quien realizó el programa* consistió en comprobar el funcionamiento de cada bloque del programa de la siguiente manera:

1. Se ejecutó el programa variando el dato inicial correspondiente al número máximo de individuos de la población a evaluar, con el fin de comprobar la capacidad de memoria de la computadora, encontrándose que soporta desde 1 hasta 200 individuos, además de que teóricamente el resultado esperado es a mayor número de individuos mayor probabilidad tendrá el programa para lograr su objetivo, esto está limitado, en el sentido de que la población inicial se genera de manera aleatoria.

2. Se probó la función de evaluación.

El módulo que si no es el más importante, si es uno de los que definen el éxito del programa, éste es el módulo de evaluación de la población, este se entiende como la función que la naturaleza realiza para decidir que individuos sobreviven y que individuos mueren. Esta función de evaluación debe definirse sobre la base de las características propias de cada problema y determinar que condiciones intervendrán para evaluar si un

individuo es apto o no lo es, es por ello, que si la función de evaluación no toma los parámetros necesarios y suficientes, el programa dará resultados poco satisfactorios, de ahí su importancia. La función de evaluación del programa de asignación de horarios tuvo diferentes pruebas con diferentes opciones, a continuación se explican algunas de ellas.

- En la evaluación de cada individuo se utilizó una nueva rutina que hace la función de cruza y mutación, la cual modifica aquellas características de los cromosomas que lo hacen ser no apto y así producir hijos que son mejores individuos que los padres, entendiéndose como cruza que dos o más características son modificadas y mutación cuando una única característica es modificada.

- La evaluación de cada individuo se hace comparando cada parte del cromosoma que representa una característica que debe coincidir con las condiciones que presenta el problema en cada momento del proceso, para identificar las diferentes características que componen cada cadena, se asigna una clave compuesta por un número de bits igual al número de características del individuo. Si una característica del individuo corresponde con la condición del problema entonces se le asigna el valor de 1 al bit que corresponda a esta característica, en caso contrario se le asigna un 0 a este bit.

- El complemento de esta función de evaluación es el procedimiento que se encarga de tomar la clave de cada individuo evaluado y aplicar el operador de cruza y mutación a través de la función de variación que es una forma de cruza y mutación, porque varía la característica completa o un bit en la característica no apta.

Al probarse este módulo, los resultados fueron positivos, ya que se obtuvieron individuos aceptables, encontrando un nuevo error en el programa, no terminaba correctamente la asignación de cada hora clase de todas las materias, repitiendo algunas asignaciones o pasando por alto otras.

Con este nuevo error, la siguiente tarea fue hacer un análisis más detallado de las condiciones que intervienen en la función de evaluación y que cada individuo debía cumplir para ser aceptado, mejorando las condiciones y los procesos de búsqueda y comparación entre los individuos de la población tanto a evaluar como la formada por individuos válidos, mejorando las funciones de asignación de laboratorios y modificando los límites en la duración de los horarios en los diferentes turnos, el programa mostró un mejor funcionamiento, alcanzando el objetivo de colocar las horas clase de cada materia a asignar en cada grupo, semestre, carrera y turno.

Al ejecutar el programa, se obtuvieron asignaciones de horas solapadas, lo que se resolvió mediante la corrección en el proceso lista, pues no detectaba la asignación en la misma hora del mismo día, con esta corrección el programa obtuvo una asignación de materias dentro de los límites de cada turno.

Se detectó que el programa asigna las horas clase de cada materia en forma desordenada, es decir, si una materia tiene diez horas clase a la semana, normalmente se colocan dos horas clase al día una seguida de la otra, para corregir esta situación se

modificó la función de copiar datos, de tal forma que se asignan tantas horas clase al día según corresponda a cada materia.

5.3. Prueba del cliente al programa

La ***Prueba del cliente al programa*** se efectuó en dos casos, cada uno con diferente presentación de resultados.

El primer caso consistió en una presentación de resultados en una pantalla similar a la que presenta el sistema operativo MS-DOS (fondo negro y un cursor esperando instrucciones o alguna respuesta), en ella se muestran los mensajes e instrucciones para el usuario (figura 5.1).

Figura. 5.1. *Presentación de resultados tipo pantalla de MS-DOS*

El segundo caso consistió en una presentación gráfica, que muestra una tabla similar a la que elabora el personal administrativo del plantel, se muestra una barra de menú con las posibles opciones de consulta de resultados, el usuario interactúa con el programa con la ayuda del mouse (ver figura 5.2).

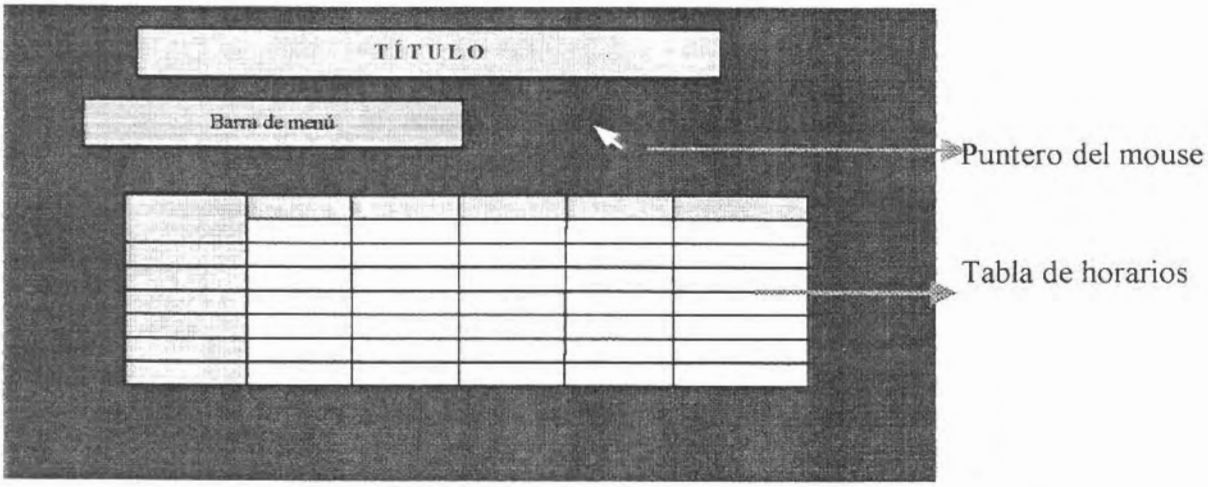


Figura. 5.2. *Presentación de resultados con gráficos y mouse*

La siguiente prueba del programa fue hacer una comparación de las tablas de asignación que realiza el personal administrativo del plantel con las tablas de asignación que realiza el sistema

5.4. Resultados

De estas pruebas se obtuvieron los siguientes resultados:

- El usuario prefiere un mouse para interactuar con el programa en lugar de utilizar sólo el teclado.

- Resulta más agradable para el usuario observar una pantalla con gráficos a colores que observar una pantalla negra y mensajes a lo ancho de ella.

- El usuario prefiere realizar la consulta de las tablas de horarios de clase con un clic del mouse, a realizar la búsqueda manual de la tabla que se desee.

- Las actividades de modificación de tablas de horarios de clase las prefiere hacer en pantalla y por medio del mouse a realizarlas en las hojas de papel, porque el programa realiza la búsqueda y comparación de datos entre las tablas, tareas que para el personal administrativo son tediosas de realizar.

- De la prueba de comparación de resultados obtenidos por el sistema con los resultados obtenidos manualmente por las personas del plantel, encargadas de realizar dicha actividad, se obtuvieron los siguientes resultados:

    - El programa asignó correctamente el total de materias, de horas clase y dentro del límite del turno correspondiente a cada grupo.

    - El orden en que se asignaron las materias que tienen horas clase mayor a 10, cumple con el orden que el personal desea y realiza.

    - Se observaron diferencias en el número de grupo que le asigna el sistema y el que le asigna el personal administrativo a cada grupo. El personal administrativo asigna como clave del número de grupo, sólo el semestre, la carrera a la que pertenece dicho grupo y un número consecutivo de grupos pertenecientes al mismo semestre, de tal forma que el grupo que el sistema asigna 1111, el personal administrativo le asigna 1104 ó 1105 según el número de grupo correspondiente a la carrera de informática.

    - El personal administrativo hizo una observación importante, esta se refiere a el número asignado a cada carrera, el programa toma como carrera número 1 a informática, como número 2 a contabilidad y como número 3 a la carrera de administración, el personal administrativo del plantel lleva el control en orden inverso, es decir, la carrera número 1 para ellos es administración, la carrera número 2 es contabilidad y la 3 es informática.

- Con respecto a la asignación de horarios de clase en los laboratorios de cómputo, se asignaron las materias correspondientes en el orden que lo realiza el personal administrativo. Se le dio el mejor uso a las horas de laboratorio disponibles, para poder obtener una asignación aceptable y dentro de los límites de cada turno.

- En la asignación de horarios de clase para los profesores se presentan las siguientes observaciones:
  - Existen profesores con una carga horaria dentro del límite que marca la institución.
  - La asignación de materias está dentro del área de cada profesor.
  - El programa asigna una carga horaria a cada profesor, en cada área según el orden en que se encuentran registrados en la lista que utiliza el programa, de tal manera, que se tienen profesores hasta con 20 horas clase a la semana y otros con 3 ó 5 horas clase siendo de la misma área. El personal administrativo trata de asignar un número de horas clase a profesores de la misma área y turno de manera equitativa.

Las tablas de horario de cada grupo elaboradas por el personal administrativo se localizarán en el apéndice 2 al final del documento. A continuación se presentan ejemplos de algunas tablas de horarios que se obtienen al ejecutar el programa.

5.5. Ejemplos

El programa puede realizar la impresión de las tablas de horarios de clase que se deseen, como se puede ver en los siguientes ejemplos.

Cada tabla es una asignación de un grupo, un laboratorio y de un profesor, son sólo una muestra de cómo asigna el programa, al ejecutarse el programa se podrá imprimir la tabla que se desee.

Tabla de horario del grupo: 1111

Hora	Lunes	Martes	Miercoles	Jueves	Viernes
7:00-8:00	amb_graf	amb_graf	comunicac	matematics	estruc_com
8:00-9:00	amb_graf	amb_graf	comunicac	matematics	matematics
9:00-10:00	matematics	comunicac	ingles	ingles	matematics
10:00-11:00	computac	comunicac	ingles		computac
11:00-12:00	valores		valores		computac
12:00-13:00	valores	oper_sist	valores		oper_sist
13:00-14:00		oper_sist	estruc_com	estruc_com	oper_sist
14:00-15:00			estruc_com	estruc_com	

Tabla 5.1. *Asignación de horarios a grupos*

La tabla 5.1 representa la asignación de horario correspondiente al grupo 1111, como se mencionó anteriormente, este número indica el semestre, el turno, la carrera y el número consecutivo de grupos de la misma carrera y semestre, en ella se muestran los días, las horas y las materias asignadas.

Tabla de horario del laboratorio: 1 turno: Matutino

Hora	Lunes	Martes	Miercoles	Jueves	Viernes
7:00-8:00	1114	1114	1114	1114	1114
8:00-9:00	1114		1114	1114	1114
9:00-10:00	1114	1114	1114	1114	1114
10:00-11:00	1114	1114	1114		1114
11:00-12:00	1113	1113	1113	1113	1113
12:00-13:00	1113	1113	1113	1113	1113
13:00-14:00	1112	1112	1112	1112	1112
14:00-15:00			1111		1111

Tabla 5.2. *Asignación de horarios a laboratorios de cómputo*

La tabla 5.2 muestra la asignación realizada por el programa al laboratorio de cómputo número 1 del turno matutino, se muestran las horas, los días y el número de cada grupo que ocupará la hora asignada.

Tabla de horario del profesor: Miguel_Angel turno: Matutino

Hora	Lunes	Martes	Miercoles	Jueves	Viernes
7:00-8:00		matematics	matematics	matematics	matematics
8:00-9:00		matematics	matematics	matematics	matematics
9:00-10:00			matematics	matematics	matematics
10:00-11:00				matematics	
11:00-12:00			matematics		
12:00-13:00			matematics		
13:00-14:00					
14:00-15:00					

Tabla 5.3. *Asignación de horarios a profesores*

La tabla 5.3 muestra la asignación realiza al profesor cuyo nombre se indica en la parte superior derecha, además se pueden observar las horas, los días y las materias que se le asignaron

5.6. Conclusiones

El tiempo empleado por el programa es menor en comparación con el empleado por el personal administrativo del plantel. El programa realiza la asignación de horarios en unos segundos y el personal administrativo dedica una semana trabajando de tres a cinco horas diarias para obtener las tablas de horarios de cada grupo, laboratorio de cómputo y de los profesores.

El manejo de las tablas de cada grupo y laboratorio es más fácil de realizar por medio de la computadora que manualmente. La utilización del mouse y una gráfica a colores en la pantalla facilita la consulta de tablas de horarios. La presentación de resultados es atractiva y sencilla para el personal encargado de realizar dicha asignación de horarios en el plantel.

CONCLUSIONES

Se realizó un análisis de los métodos de optimización que tradicionalmente se usan en la búsqueda de soluciones, mostrando que estos obtienen resultados no óptimos cuando el espacio de búsqueda es muy grande, es decir el número de parámetros a encontrar es grande, principalmente porque al resolver derivadas de orden mayor se obtienen errores de redondeo y muestran una fuerte dependencia del punto de partida.

Algoritmos genéticos se propuso como una alternativa para resolver este tipo de problemas, principalmente por la robustez y el paralelismo implícito que presenta.

Se creo una rutina llamada *variación* que hace la función de cruza y mutación de individuos., ya que a partir de tomar algunos cromosomas (padres) poco aptos, crea nuevos cromosomas (hijos) que son mejor que sus padres.

El poder resolver este tipo de problemas que se presentan en la institución utilizando sus recursos con los que cuenta, permite que el personal encargado de realizar esta tarea pueda dedicar más tiempo a sus otras actividades, permitiéndoles realizarlas más cómodamente.

Se cumplió el objetivo propuesto por esta tesis y el programa implementado se mejoró de tal manera que permite al personal administrativo realizar la asignación de horarios de forma más sencilla.

Algunos de los problemas que se presentaron al realizar esta tarea son los siguientes:

Primero: Determinar las condiciones y restricciones que el plantel necesita para la asignación de horarios e implementarlas en el programa, de tal forma que el algoritmo genético trabaje con ellas.

Segundo: se implementó una nueva rutina de cruza y mutación, la cual compara las características de cada individuo con las condiciones y restricciones del problema, las características que no cumplen con las condiciones del problema se varian, dejando fijas aquellas que si cumplen estas condiciones.

Para verificar la viabilidad de la propuesta se realizaron las tablas de horarios por medio del programa y se compararon con las tablas obtenidas por el personal administrativo del plantel, obteniéndose resultados aceptables.

En el transcurso de la elaboración del programa se presentó ante el personal administrativo encargado de la asignación de horarios del plantel, los que a su vez hicieron sus comentarios y sugerencias del mismo, lo que permitió realizar mejoras al programa, entre ellas la manera de presentar los resultados en pantalla y la opción para modificar algunas asignaciones realizadas.

El tiempo que invierte el personal administrativo al realizar las asignaciones de horarios de manera manual es de una semana, esto en comparación con minutos o segundos que ocupa el algoritmo implementado es demasiado tiempo y es una ventaja del programa que se implementó.

En la elaboración del programa se utiliza memoria dinámica, al emplear listas ligadas, esto facilita el manejo de memoria, es decir, la memoria que no se utilice por el programa, se devolverá al sistema operativo y a su vez si el programa requiere memoria, esta se reserva con instrucciones especificas en la codificación del programa.

Las estructuras utilizadas en el programa se componen de campos de bits, este tipo de campos permite utilizar memoria en la medida que cada campo de la estructura lo requiera, es decir, para guardar el valor 0 ó 1 solo requiere un bit y no los 16 que normalmente se utilizan para almacenar un valor entero cualquiera.

Se proponen como trabajo futuro, refiriéndose exclusivamente a la asignación de horarios en el plantel CONALEP las siguientes actividades:

Realizar las modificaciones pertinentes al programa de tal forma que sea aplicable no sólo en instituciones como CONALEP sino en cualquier institución educativa que requiera de facilitar y agilizar la asignación de sus horarios respectivos.

Se propone, además de lo anterior, implementar este tipo de algoritmo en un lenguaje diferente a lenguaje C, con el objetivo de mejorar la interfaz y hacer de este programa un proyecto más atractivo.

Considerar de manera general todos los semestres al mismo tiempo y no por periodo como fue el caso, ya que en algunas instituciones se imparten clases a semestres pares e impares.

Esperamos que este trabajo sea una muestra de la aplicación de algoritmos genéticos.

Referencias

[1] David E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine learning*, The University of Alabama, Addison-Wesley Publishing company, Inc., 2-5.

[2] *Diseño de sistemas ópticos usando algoritmos genéticos*, Sergio Vázquez y Montiel, Instituto Nacional de Astrofisica Óptica y Electrónica, Tonantzintla, Puebla, México, 14-15.

[3] David E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, The University of Alabama, Addison-Wesley Publishing company, Inc., 3-4.

[4] David E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, The University of Alabama, Addison-Wesley Publishing company, Inc., 3-6.

[5] N. A Rosenbluth, M. Rosenbluth, A. Teller and E. Teller, *Equation of state calculations by fast machines*, Journal of Chemical Physics, 1087-1092

[6] Robert Eisberg, Robert Resnick, *Física cuántica*, Universidad de California, Editorial Limusa, 30-33.

[7] Herbert Schildt, *C . Manual de referencia*, Segunda edición, Osborne/Mc graw hill, Madrid, 1990, 130-135.

[8] Herbert Schildt, *C. Manual de referencia*, Segunda edición, Osborne/Mc graw hill, Madrid, 1990, 130-135.

[9] David E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, The University of Alabama, Addison-Wesley Publishing company, Inc., 3-7.

[10] David E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, The University of Alabama, Addison-Wesley Publishing company, Inc., 3-7.

[11] Sergio Vázquez y Montiel, *Diseño de sistemas ópticos usando algoritmos genéticos*, Instituto Nacional de Astrofisica Óptica y Electrónica, Tonantzintla, Puebla, México, 25-30.

[12] Sergio Vázquez y Montiel, *Diseño de sistemas ópticos usando algoritmos genéticos*, Instituto Nacional de Astrofisica Óptica y Electrónica, Tonantzintla, Puebla, México, 25-30.

[13] David E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, The University of Alabama, Addison-Wesley Publishing company, Inc., 10-17.

[14] Sergio Vázquez y Montiel, *Diseño de sistemas ópticos usando algoritmos genéticos*, Instituto Nacional de Astrofisica Óptica y Electrónica, Tonantzintla, Puebla, México, 25-35.

[15] David E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, The University of Alabama, Addison-Wesley Publishing company, Inc., 10-17.

[16] Sergio Vázquez y Montiel, *Diseño de sistemas ópticos usando algoritmos genéticos*, Instituto Nacional de Astrofisica Óptica y Electrónica, Tonantzintla, Puebla, México, 25-35.

[17] Sergio Vázquez y Montiel, *Diseño de sistemas ópticos usando algoritmos genéticos*, Instituto Nacional de Astrofisica Óptica y Electrónica, Tonantzintla, Puebla, México, 25-35.

[18] Sergio Vázquez y Montiel, *Diseño de sistemas ópticos usando algoritmos genéticos*, Instituto Nacional de Astrofisica Óptica y Electrónica, Tonantzintla, Puebla, México, 25-35.

[19] David E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, The University of Alabama, Addison-Wesley Publishing company, Inc., 10-17.

[20] David E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, The University of Alabama, Addison-Wesley Publishing company, Inc., 10-17.

[21] David E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, The University of Alabama, Addison-Wesley Publishing company, Inc., 10-17.

[22] Herbert Schildt, *C . Manual de referencia*, Segunda edición, Osborne/Mc graw hill, Madrid, 1990, 130-135.

APÉNDICE 1

El archivo conmouse.cpp contiene la función principal, en donde se inicia la ejecución del programa.

```
# include <dos.h>
# include <conio.h>
# include <stdio.h>
# include <stdlib.h>
# include <string.h>
# include <graphics.h>
# include <c:\borlandc\bin\mouse.cpp>
# include <c:\borlandc\bin\horass.cpp>
# include <c:\borlandc\bin\mygraf.cpp>
# include <c:\borlandc\bin\flechas.cpp>
# include <c:\borlandc\bin\move_bar.cpp>
# include <c:\borlandc\bin\opciones.cpp>
# include <c:\borlandc\bin\modifica.cpp>
void main()  //funcion principal
{
horarios();
grafico();
opcion();
}

/*--- archivo archivos.cpp ----------------------------------------------------------------------------------------------*/
            Este archivo captura los datos de entrada al programa y genera la población inicial, contiene las rutinas de evaluación y variación de los
            individuos de la población.
# include <c:\horarios\leedatos.cpp>
# include <c:\horarios\pobnue.cpp>

void evalua(),f_evalua(),horarios();
int conta,bit,mvalue;
unsigned value1,value2;

void tamano()
{
char *str="abcdefghij";

do{
        do{
                gotoxy(1,10);
                clreol();
                gets(str);
                pob=atoi(str);
                }while(pob<=0);
        }while(pob>200);
}

void mutacion()
{
        if(value==125)
                {bit=rand() % 3;
                mvalue=as1->dia;
                for(j=0;j<bit;j++)
                        mvalue>>=1;
                if(mvalue & mask)
                        as1->dia=as1->dia-pow(2,bit);
                else
                        as1->dia=as1->dia+pow(2,bit);
                }
        else
                {if(value==123)
                        {bit=rand() % 5;
                        mvalue=as1->hora;
                        for(j=0;j<bit;j++)
                                mvalue>>=1;
                        if(mvalue & mask)
                                as1->hora=as1->hora-pow(2,bit);
                        else
                                as1->hora=as1->hora+pow(2,bit);
                        }
                else
                        {if(value==119)
                                {bit=0;
                                mvalue=as1->turno;
                                for(j=0;j<bit;j++)
                                        mvalue>>=1;
                                if(mvalue & mask)
                                        as1->turno=as1->turno-pow(2,bit);
```

```
                                        else
                                                as1->turno=as1->turno+pow(2,bit);
                                        }
                                else
                                        {if(value==101)
                                                {bit=rand() % 2;
                                                mvalue=as1->carrera;

                                                for(j=0;j<bit;j++)
                                                        mvalue>>=1;
                                                if(mvalue & mask)
                                                        as1->carrera=as1->carrera-pow(2,bit);
                                                else
                                                        as1->carrera=as1->carrera+pow(2,bit);
                                                }
                                        else
                                                {if(value==95)
                                                        {bit=rand() % 3;
                                                        mvalue=as1->semestre;
                                                        for(j=0;j<bit;j++)
                                                                mvalue>>=1;
                                                        if(mvalue & mask)
                                                                as1->semestre=as1->semestre-pow(2,bit);
                                                        else
                                                                as1->semestre=as1->semestre+pow(2,bit);
                                                        }
                                                else
                                                        {if(value==63)
                                                                {bit=rand() % 6;
                                                                mvalue=as1->materia;
                                                                for(j=0;j<bit;j++)
                                                                        mvalue>>=1;
                                                                if(mvalue & mask)
                                                                        as1->materia=as1->materia-pow(2,bit);
                                                                else
                                                                        as1->materia=as1->materia+pow(2,bit);
                                                                }
                                                        }
                                                }
                                        }
                                }
                        }
}

void horarios()
{
int temp;
randomize();
clrscr();
printf("\n\n  Teclee el numero del periodo que corresponda a la actual\n");
printf("       distribucion de horarios...\n");
printf("1. Septiembre-Febrero\n");
printf("2. Marzo-Agosto\n");
inic();
printf("\n\nTamaño de la poblacion a generar (mayor que 0 ) :");
tamano();
funcion_lee_datos();
pob_inic();


printf("\n");
cuennodos=0;
cuennodosmateria=0;
while(teorias==0)
        {
        t0lab1=35;
        t0lab2=35;
        t1lab1=35;
        t1lab2=35;
        teorias=0;
        for(tt1=tt;tt1!=NULL;tt1=tt1->sig)
                {
                if(reinicio==1)
                        {
                        tt1=tt;
                        printf("R");
                        reinicio=0;
                        }
                gruposs=tt1->grupos;
                if(tt1->turno==0)
```

```
                horass=7;
        else
                horass=14;
        diass=1;
        while(gruposs>0)
                {
                for(m1=m;m1!=NULL;)
                        {
                        if(m1->hrs_prac!=0 && (m1->carrera==tt1->carrera || m1->carrera==0)
                                && m1->semestre==tt1->semestre && m1->asigna==0)
                                {break;}
                        else
                                m1=m1->sig;
                        }
                if(m1!=NULL)
                        {nodos=m1->hrs_prac+m1->hrs_teo;//tt1->hrs_grupo;
                        cuennodos=0;
                        while(cuennodos<nodos)
                                {
                                para_variar++;
                                if(para_variar>=100000)
                                        reinit();
                                else
                                        {
                                        evalua();
                                        if(reinicio==1)
                                                break;
                                        if(cuennodos!=nodos)
                                        varia();
                                        }
                                }
                        }

                else
                        gruposs=0;
                if(reinicio==1)
                        break;
                }
        printf("Ü");//²");
        }

if(tt1==NULL)
        {
        practicas=1;
        for(tt1=tt;tt1!=NULL;tt1=tt1->sig)
                {
                if(reinicio==1)
                        {
                        tt1=tt;
                        printf("t");
                        reinicio=0;
                        if(tt1->turno==0)
                                horass=7;
                        else
                                horass=14;
                        diass=1+rand()%5;
                        }
                diass=1+rand()%5;
                gruposs=tt1->grupos;
                while(gruposs>0)
                        {
                        diass=1+rand()%5;
                        for(m1=m;m1!=NULL;)
                                {
                                if(m1->hrs_teo!=0 && m1->hrs_prac==0 && (m1->carrera==tt1->carrera || m1->carrera==0)
                                        && m1->semestre==tt1->semestre && m1->asigna==0)
                                        {break;}
                                else
                                        m1=m1->sig;
                                }
                        if(m1!=NULL)
                                {nodos=m1->hrs_teo;//tt1->hrs_grupo;
                                cuennodos=0;
                                while(cuennodos<nodos)
                                        {
                                        para_variar++;
                                        if(para_variar>=100000)
                                                reinit();
                                        else
                                                {
```

```
                                                                        evalua();
                                                                        if(reinicio==1)
                                                                                break;
                                                                        if(cuennodos!=nodos)
                                                                                varia();
                                                                        }
                                                                }
                                                        }
                                                else
                                                        {
                                                        gruposs--;
                                                        for(m1=m;m1!=NULL;m1=m1->sig)
                                                                m1->asigna=0;
                                                        }
                                                if(reinicio==1)
                                                        break;
                                                }
                                        printf("Ût");//²");
                                        if(reinicio==1)
                                                break;
                                        }
                                if(reinicio==1)
                                        teorias=0;
                                else
                                        teorias=1;
                                }
                        }

for(d1=d;d1!=NULL;d1=d1->sig)
                        d1->horas=0;

for(as3=as2;as3!=NULL;as3=as3->sig)
                        {
                        for(d1=d;d1!=NULL;)
                                {
                                if(d1->area==as3->area && d1->turno==as3->turno && d1->horas<20)
                                        {
                                        as3->profe=d1->profe;
                                        d1->horas++;
                                        d1=NULL;
                                        }
                                else
                                        d1=d1->sig;
                                }
                        }

for(as3=as2;as3!=NULL;as3=as3->sig)
                        {
                        for(as1=as3->sig;as1!=NULL;as1=as1->sig)
                                {
                                if(as1->profe==as3->profe && as1->dia==as3->dia && as1->hora==as3->hora)
                                        {
                                        for(d1=d;d1!=NULL;)
                                                {
                                                if(d1->profe==as3->profe)
                                                        {
                                                        d1->horas=20;
                                                        for(d1=d;d1!=NULL;)
                                                                {
                                                                if(d1->area==as1->area && d1->turno==as1->turno && d1->horas<20)
                                                                        {
                                                                        as1->profe=d1->profe;
                                                                        d1->horas++;
                                                                        d1=NULL;
                                                                        as1=as3;
                                                                        }
                                                                else
                                                                        d1=d1->sig;
                                                                }
                                                        }
                                                else
                                                        d1=d1->sig;
                                                }
                                        }
                                }
                        }

for(as3=as2;as3!=NULL;as3=as5->sig)
                        {
                        for(as1=as3->sig,as5=as3;as1!=NULL;)
```

```
                        {
                        if(as1->turno==as3->turno && as1->carrera==as3->carrera &&
                                as1->semestre==as3->semestre && as1->grupo==as3->grupo)
                                {
                                if(as5->sig==as1)
                                        {
                                        as5=as1;
                                        as1=as1->sig;
                                        }
                                else
                                        {
                                        as6->sig=as1->sig;
                                        as1->sig=as5->sig;
                                        as5->sig=as1;
                                        as5=as1;
                                        as1=as1->sig;
                                        }
                                }
                        else
                                {
                                as6=as1;
                                as1=as1->sig;
                                }
                        }
        }

tt1=tt;
while(tt!=NULL)
        {
        tt=tt->sig;
        free(tt1);
        tt1=tt;
        }
free(tt1);
free(tt);
m1=m;
while(m!=NULL)
        {
        m=m->sig;
        free(m1);
        m1=m;
        }
free(m1);
free(m);
as1=as;
while(as!=NULL)
        {
        as=as->sig;
        free(as1);
        as1=as;
        }
free(as1);
free(as);
free(as4);
fclose(fp);
i=0;
}

void evalua()
{
para_variar++;
if(para_variar>=100000)
        reinit();
else
        {
        matteria=1;
        for(as1=as;as1!=NULL;as1=as1->sig)
                {
                if(cuennodos==nodos)
                        break;
                as1->fijar=0;
                conta=0;
                if(as1->semestre==tt1->semestre && as1->turno==tt1->turno)
                        {
                        as1->fijar=40;
                        f_evalua();
                        }
                else
                        as1->fijar=0;//95;
                }
```

```
		}
}

void f_evalua()
{
as1->fijar=0;
/*if(m1->semestre==tt1->semestre)
		{*/
		if(m1->carrera==0 || m1->carrera==tt1->carrera)
			{
			if(as1->materia==m1->materia)
				{
				strcpy(as1->nom,m1->nom);
				as1->fijar=(as1->fijar<<=1)|1;
				conta++;
				}
			else
				{
				as1->fijar=(as1->fijar<<=1)|0;
				}
			if(as1->semestre==m1->semestre)
				{
				as1->fijar=(as1->fijar<<=1)|1;
				conta++;
				}
			else
				{
				as1->fijar=(as1->fijar<<=1)|0;
				}
			if(as1->carrera==tt1->carrera)
				{
				as1->fijar=(as1->fijar<<=1)|1;
				conta++;
				}
			else
				{
				as1->fijar=(as1->fijar<<=1)|0;
				}
			if(as1->turno==tt1->turno)
				{
				as1->fijar=(as1->fijar<<=1)|1;
				conta++;
				}
			else
				{
				as1->fijar=(as1->fijar<<=1)|0;
				}
			if(practicas==0)
				as1->hrs_tot=m1->hrs_teo+m1->hrs_prac;
			else
				as1->hrs_tot=m1->hrs_teo+m1->hrs_prac;
			if(as1->hora/*>=7 && as1->hora<=14)//*/==horass)// && as1->hora<=14)
				{
				as1->fijar=(as1->fijar<<=1)|1;
				conta++;
				}
			else
				{
				as1->fijar=(as1->fijar<<=1)|0;
				}
			if(practicas==0)
				{
				as1->hrs_prac=m1->hrs_prac;
				if(clave_lab==0)
					asigna_lab(0);
				}
			if(cuennodos>=m1->hrs_prac)
				as1->lab=0;
			if(as1->dia==diass)//*/>=1 && as1->dia<=5)
				{
				as1->fijar=(as1->fijar<<=1)|1;
				conta++;
				}
			else
				{
				as1->fijar=(as1->fijar<<=1)|0;
				}
			as1->grupo=gruposs;//tt1->grupos;
			if(tt1!=NULL)
```

```
                                        {
                                        if(conta==6) / 7)
                                                {lista();}
                                        }
                        }
                else
                        {as1->fijar=40;}
}

void varia()
{
for(as1=as;as1!=NULL;as1=as1->sig)
                {
                value=as1->fijar;
                if(/*value==126 || */value==125 || value==123 || value==119
                        || value==101 || value==95 || value==63)
                                mutacion();
                else
                        {
                        if(value & mask)
                                {value>>=1;}
                        else
                                {
                                as1->dia=1+rand() % 5;
                                value>>=1;
                                }
                        if(value & mask)
                                {value>>=1;}
                        else
                                {
                                as1->hora=7+rand() % 16;
                                value>>=1;
                                }
                        if(value & mask)
                                {value>>=1;}
                        else
                                {
                                as1->turno=rand() % 2;
                                value>>=1;
                                }
                        if(value & mask)
                                {value>>=1;}
                        else
                                {
                                as1->carrera=1+rand() % 3;
                                value>>=1;
                                }
                        if(value & mask)
                                {value>>=1;}
                        else
                                {
                                as1->semestre=1+rand() % 6;
                                value>>=1;
                                }
                        if(value & mask)
                                {value>>=1;}
                        else
                                {
                                as1->materia=1+rand() % num_materias;
                                value>>=1;
                                }
                        }
                }
as1=as;
}
```

El archivo leedatos.cpp se encarga de revisar en la lista de individuos aceptados que no existan copias de ellos, para evitar solapamientos.

```
# include <string.h>
void funcion_lee_datos(),buscatt(int),asigna_materias();
void varia(),inic (void),pob_inic();

char opc,*array;
char
*grupos[20]={"abcd","abcd","abcd","abcd","abcd","abcd","abcd","abcd","abcd","abcd","abcd","abcd","abcd","abcd","abcd","abcd","abcd","abcd","abcd","abcd",N
ULL};
int pob;
unsigned value, mask=1>>0;
```

```
struct asignaciones{
        unsigned materia : 6;
        unsigned semestre : 3;
        unsigned turno : 1;
        unsigned dia : 3;
        unsigned hora : 5;
        unsigned grupo : 3;
        unsigned carrera:2;
        unsigned lab:2;
        unsigned hrs_tot:5;
        unsigned hrs_prac:5;
        char nom[25];
        unsigned profe :7;
        unsigned area:5;
        char nom_area[20];
        unsigned fijar : 8;
        struct asignaciones *sig;
        };

typedef struct asignaciones ASIGNA;
typedef ASIGNA *asig;

asig nodo_asigna()
{
        return((asig) malloc(sizeof(ASIGNA)));
}

ASIGNA *as=NULL,*as1=NULL,*as2=NULL,*as3=NULL,
        *as4=NULL,*as5=NULL,*as6=NULL,*as7=NULL,*ptr1=NULL,*ptr2=NULL;

struct materias{
        unsigned materia : 6;
        unsigned semestre : 3;
        unsigned hrs_teo : 4;
        unsigned hrs_prac:4;
        unsigned carrera:2;
        char nom[25];
        unsigned area:5;
        unsigned asigna:1;
        struct materias *sig;
        };
struct totales{
        unsigned semestre : 3;
        unsigned grupos : 4;
        unsigned turno : 1;
        unsigned carrera: 2;
        int hrs_tot;
        int hrs_grupo;
        unsigned mprac:4;
        unsigned mteo:4;
        struct totales *sig;
        };
struct profesores{
        unsigned profe:8;
        unsigned turno:1;
        unsigned area:5;
        unsigned horas:6;
        char nom[25];
        struct profesores *sig;
        };

struct listaprof{
unsigned profe:8;
        unsigned turno:1;
        char nom[25];
        struct listaprof *sig;
        };


typedef struct materias MATER;
typedef MATER *mat;

mat nodo_materia()
{
        return((mat) malloc(sizeof(MATER)));
}

typedef struct totales TOTS;
```

```
typedef TOTS *tots;

tots nodo_totales()
{
        return((tots)malloc(sizeof(TOTS)));
}

typedef struct profesores PROFS;
typedef PROFS *profs;

profs nodo_profesores()
{
        return((profs)malloc(sizeof(PROFS)));
}

typedef struct listaprof LISPROFS;
typedef LISPROFS *lisprofs;

lisprofs nodo_listaprof()
{
        return((lisprofs)malloc(sizeof(LISPROFS)));
}


MATER *m=NULL,*m1=NULL;
TOTS *tt=NULL,*tt1=NULL,*tquit=NULL;
PROFS *d=NULL,*d1=NULL,*d2=NULL;
LISPROFS *asprof=NULL,*asprim=NULL,*asprim1=NULL;
FILE *fp=NULL;

int num_materias,i,j,nodos,area,profe,mate,sem,hort,horp,grup,tur,carr,semestres[3];
char nom[25];

void funcion_lee_datos()
{
fp=fopen("c:\\horarios\\copymate.dat","r");
if(fp==NULL)
        {
        printf("El archivo no puede abrirse...");
        getch();
        exit(1);
        }
m=NULL;
while(!feof(fp) && fp->level!=1)
        {
        fscanf(fp,"%u%u%u%u%u%s%u",&mate,&sem,&hort,&horp,&carr,&nom,&area);
        if(sem==semestres[0] || sem==semestres[1] || sem==semestres[2])
                {
                if(m==NULL)
                        {
                        m=nodo_materia();
                        if(m==NULL)
                                {
                                printf("\nMemoria agotada");
                                getch();
                                exit(0);
                                }
                        m1=m;
                        }
                else
                        {
                        m1->sig=nodo_materia();
                        m1=m1->sig;
                        }
                m1->materia=mate;
                m1->semestre=sem;
                m1->hrs_teo=hort;
                m1->hrs_prac=horp;
                m1->carrera=carr;
                for(i=0;i<=25;++i){m1->nom[i]=nom[i];}
                m1->area=area;
                num_materias++;
                m1->asigna=0;
                }
        }
fclose(fp);
m1->sig=NULL;

fp=fopen("c:\\horarios\\TOTALES.DAT","r");
if(fp==NULL)
```

```
                {
                printf("El archivo no puede abrirse...");
                getch();
                exit(1);
                }
tt=NULL;
while(!feof(fp) && fp->level!=1)
                {
                fscanf(fp,"%u%u%u%u",&sem,&grup,&tur,&carr);
                if(grup!=0)
                                {
                                if(sem==semestres[0] || sem==semestres[1] || sem==semestres[2])
                                                {
                                                if(tt==NULL)
                                                                {
                                                                tt=nodo_totales();
                                                                if(tt==NULL)
                                                                                {
                                                                                printf("\nMemoria agotada");
                                                                                getch();
                                                                                exit(0);
                                                                                }
                                                                tt1=tt;
                                                                }
                                                else
                                                                {
                                                                tt1->sig=nodo_totales();
                                                                tt1=tt1->sig;
                                                                }
                                                tt1->grupos=grup;
                                                tt1->semestre=sem;
                                                tt1->turno=tur;
                                                tt1->carrera=carr;
                                                tt1->hrs_tot=0;
                                                tt1->hrs_grupo=0;
                                                tt1->mprac=0;
                                                tt1->mteo=0;
                                                }
                                }
                }
fclose(fp);
tt1->sig=NULL;

fp=fopen("c:\\horarios\\PROFES.DAT","r");
if(fp==NULL)
                {
                printf("El archivo no puede abrirse...");
                getch();
                exit(1);
                }
d=NULL;
while(!feof(fp))// && fp->level!=1)
                {
                fscanf(fp,"%u%u%u%s",&profe,&tur,&area,&nom);
                if(d==NULL)
                                {
                                d=nodo_profesores();
                                if(d==NULL)
                                                {
                                                printf("\nMemoria agotada");
                                                getch();
                                                exit(0);
                                                }
                                d1=d;
                                                }
                else
                                {
                                d1->sig=nodo_profesores();
                                if(d1->sig==NULL)
                                                {
                                                printf("\nMemoria agotada");
                                                getch();
                                                exit(0);
                                                }
                                d1=d1->sig;
                                }
                d1->profe=profe;
                d1->turno=tur;
                d1->area=area;
                d1->horas=0;
```

```
            for(i=0;i<=25;++i){d1->nom[i]=nom[i];}
            }
fclose(fp);
d1->sig=NULL;


i=0;
for(tt1=tt;tt1!=NULL;tt1=tt1->sig)
            {
            if(tt1->grupos!=0)
                        {
                        for(j=0;j<tt1->grupos;j++,i++)
                                    {
                                    //pob=(tt1->semestre*1000)+((tt1->turno+1)*(100))+(tt1->carrera*10)+j+1;
                                    array=fcvt((tt1->semestre*1000)+((tt1->turno+1)*(100))+(tt1->carrera*10)+j+1,0,0,0);
                                    strcpy(grupos[i],array);
                                    }
                        tt1->hrs_tot=0;
                        for(m1=nx;m1!=NULL;)
                                    {
                                    if(m1->semestre==tt1->semestre)
                                                {
                                                if(m1->carrera==0)
                                                            {
                                                            asigna_materias();
                                                            tt1->hrs_tot=tt1->hrs_tot+((m1->hrs_teo+m1->hrs_prac)*tt1->grupos);
                                                            m1=m1->sig;
                                                            }
                                                else
                                                            {
                                                            if(m1->carrera==tt1->carrera)
                                                                        {
                                                                        asigna_materias();
                                                                        tt1->hrs_tot=tt1->hrs_tot+((m1->hrs_teo+m1->hrs_prac)*tt1->grupos);
                                                                        m1=m1->sig;
                                                                        }
                                                            else
                                                                        {
                                                                        m1=m1->sig;
                                                                        }
                                                            }
                                                }
                                    else
                                                {
                                                m1=m1->sig;
                                                }
                                    }
                        nodos=nodos+tt1->hrs_tot;
                        tt1->hrs_grupo=tt1->hrs_tot/tt1->grupos;
                        tt1->hrs_tot=tt1->hrs_grupo;
                        }
            j=i;
            }
}

void asigna_materias()
{
if(m1->hrs_prac!=0)
            {
            tt1->mprac++;
            }
else
            {tt1->mteo++;}
}

void inic()
{
gotoxy(1,7);
opc=getche();
switch(opc){
            case '1':semestres[0]=1;semestres[1]=3;semestres[2]=5;break;
            case '2':semestres[0]=2;semestres[1]=4;semestres[2]=6;break;
            default:inic();break;
            }
}

void pob_inic()
{
for(i=1;i<=pob;++i)
            {
```

```
        if(i==1){as=nodo_asigna();as1=as;}
        else
                {
                as1->sig=nodo_asigna();
                if(as1->sig==NULL)
                        {
                        printf("Memoria insuficiente\n");
                        getch();
                        exit(1);
                        }
                as1=as1->sig;
                }
        as1->materia=1+rand()%37;
        as1->semestre=1+rand()%6;
        as1->turno=rand()%2;
        as1->dia=1+rand()%5;
        as1->hora=7+rand()%16;
        as1->grupo=1+rand()%8;
        as1->carrera=1+rand()%3;
        as1->lab=0;
        as1->hrs_tot=0;
        as1->profe=1+rand()%110;
        as1->fijar=0;
        }
as1->sig=NULL;
}
```

El archivo pobnue.cpp se encarga de revisar en la lista de individuos aceptados que no existan copias de ellos, para evitar solapamientos.

```
void lista(),copia(int),f_rompe(),f_continua(),
                f_dia(),f_labo(),f_dismin(),
                asigna_lab(int),ordena_nodos();
int matteria=0,nodhechos,cuennodos=0,cuennodosmateria=0,
                t0lab1=35,t0lab2=35,t1lab1=35,t1lab2=35,ciclos=0,gen=0,
                semestre,carrera,grupo,remhora,modif=0,practicas=0,ciclodia,ciclohora,
                indic=0,practics=0,horass,diass,colocadas=0,reinicio=0,copiado=0,
                ii,horascont,hhora,matee=0,clave_lab=0,gruposs,teorias=0;
float para_variar=0;

unsigned char exist;

void reinit()
{
tt1=tt;
                                        while(tt!=NULL)
                                                {
                                                tt=tt->sig;
                                                free(tt1);
                                                tt1=tt;
                                                }
                                        free(tt1);
                                        free(tt);
                                        m1=m;
                                        while(m!=NULL)
                                                {
                                                m=m->sig;
                                                free(m1);
                                                m1=m;
                                                }
                                        free(m1);
                                        free(m);
                                        d1=d;
                                        while(d!=NULL)
                                                {
                                                d=d->sig;
                                                free(d1);
                                                d1=d;
                                                }
                                        free(d1);
                                        free(d);
                                        as1=as;
                                        while(as!=NULL)
                                                {
                                                as=as->sig;
                                                free(as1);
                                                as1=as;
```

```
                                        }
                                free(as1);
                                free(as);
                                for(as3=as2;as2!=NULL;)
                                        {as2=as2->sig;
                                        free(as3);
                                        as3=as2;
                                        }
                                free(as3);
                                free(as2);
                                as2=NULL;
                                funcion_lee_datos();
                                tt1=tt;
                                nodos=0;
                                cuennodos=0;
                                if(tt1->turno==0)
/*                                      horass=7;
                                else
                                        horass=14;
                                diass=1;*/

                                for(m1=m;m1!=NULL;m1=m1->sig)
                                        m1->asigna=0;
                                pob=1;
                                pob_inic();
                                t0lab1=35;
                                t0lab2=35;
                                t1lab1=35;
                                t1lab2=35;
                                matee=0;
                                practics=0;
                                ciclos=0;
                                para_variar=0;
                                gruposs=0;reinicio=1;
                                practicas=0;clave_lab=0;
}

void funcion_hace_nodo_y_copia(int hhora)
{
if(cuennodos>=m1->hrs_prac)
        as1->lab=0;
if(as2==NULL)
        {as2=nodo_asigna();
        as3=as2;
        }
else
        {as3->sig=nodo_asigna();
        as3=as3->sig;
        }
copia(hhora);
}

void lista1(int horaa)
{
if(as2==NULL)
        {
        exist=0;
        }
else
        {
        for(as4=as2;as4!=NULL;)
                {if(as4->turno==as1->turno)
                        {if(as4->semestre==as1->semestre)
                                {if(as4->carrera==as1->carrera)
                                        {if(as4->grupo==as1->grupo)
                                                {if(as4->materia==as1->materia)
                                                        {if(as4->dia==as1->dia)
                                                                {if(as4->hora==horaa)//as1->hora)
                                                                        {exist=1;as4=NULL;}
                                                                else
                                                                        {exist=1;as4=NULL;}
                                                                }
                                                        else
                                                                {
                                                                as1->hrs_tot--;
                                                                if(as1->hrs_prac>=1)
                                                                        as1->hrs_prac--;
                                                                if(as1->hrs_tot==0)
```

```
                                                                                {m1-
>asigna=1;matee=1;exist=1;as4=NULL;}
                                                                                else
                                                                                {
                                                                                exist=0;as4=as4->sig;
                                                                                }
                                                                        }
                                                                }
                                                        else
                                                                {if(as4->dia==as1->dia)
                                                                        {if(as4->hora==horaa)//as1->hora)
                                                                                {exist=1;as4=NULL;}
                                                                        else
                                                                                {exist=0;as4=as4->sig;}
                                                                        }
                                                                else
                                                                        {exist=0;as4=as4->sig;}
                                                                }
                                                        }
                                                else
                                                        {
                                                        if(as4->lab==as1->lab && as1->lab>0)
                                                                {if(as4->dia==as1->dia)
                                                                        {if(as4->hora==horaa)//as1->hora)
                                                                                {exist=1;as4=NULL;}
                                                                        else
                                                                                {exist=0;as4=as4->sig;}
                                                                        }
                                                                else
                                                                        {exist=0;as4=as4->sig;}
                                                                }
                                                        else
                                                                {exist=0;as4=as4->sig;}
                                                        }
                                                }
                                        else
                                                {
                                                if(as4->lab==as1->lab && as1->lab>0)
                                                        {if(as4->dia==as1->dia)
                                                                {if(as4->hora==horaa)//as1->hora)
                                                                        {exist=1;as4=NULL;}
                                                                else
                                                                        {exist=0;as4=as4->sig;}
                                                                }
                                                        else
                                                                {exist=0;as4=as4->sig;}
                                                        }
                                                else
                                                        {exist=0;as4=as4->sig;}
                                                }
                                        }
                                else
                                        {
                                        if(as4->lab==as1->lab && as1->lab>0)
                                                {if(as4->dia==as1->dia)
                                                        {if(as4->hora==horaa)//as1->hora)
                                                                {exist=1;as4=NULL;}
                                                        else
                                                                {exist=0;as4=as4->sig;}
                                                        }
                                                else
                                                        {exist=0;as4=as4->sig;}
                                                }
                                        else
```

```
                                        {exist=0;as4=as4->sig;}
                                }
                        }
                }
}
void funcion_horas()
{
horass++;
if(tt1->turno==0)
        {if(horass<13)
                {diass++;
                if(diass<5)
                        {diass=1;
                        ciclos++;
                        if(ciclos==3)
                                {
                                if(as1->hrs_prac!=0)
                                        {if(t0lab2>0)
                                                {
                                                as1->lab=2;
                                                clave_lab=1;
                                                ciclos=0;
                                                }
                                        }
                                else
                                        {
                                        reinit();
                                        }
                                }
                        }
                horass=7;
                }
        }
else
        {
        if(horass>20)
                {diass++;
                if(diass>5)
                        {diass=1;
                        ciclos++;
                        if(ciclos==3)
                                {
                                ciclos=0;
                                if(as1->hrs_prac!=0)
                                        {if(t1lab2>0)
                                                {
                                                as1->lab=2;
                                                clave_lab=1;
                                                ciclos=0;
                                                }
                                        }
                                else
                                        {
                                        reinit();
                                        }
                                }
                        }
                horass=14;
                }
        }
}

void lista()
{
int hortot,horprac;
hortot=as1->hrs_tot;
horprac=as1->hrs_prac;
lista1(as1->hora);
if(exist==0)
        {
        if((m1->hrs_teo+m1->hrs_prac)<=10)
                {
                if(as1->turno==0)
                        {if(as1->hora<14)
                                {if(as1->hrs_tot==1)
                                        {funcion_hace_nodo_y_copia(as1->hora);
                                        m1->asigna=1;
                                        if(clave_lab==1)
                                                {
```

```
                                        as4=as1;
                                        for(as1=as;as1!=NULL;as1=as1->sig)
                                                as1->lab=1;
                                        as1=as4;
                                        }
                                clave_lab=0;
                                diass++;
                                if(diass>5)
                                        diass=1;
                                horass=7;
                                }
                        else
                                {
                                as1->hrs_tot=hortot;
                                as1->hrs_prac=horprac;
                                lista1(as1->hora+1);
                                if(exist==0)
                                        {
                                        if(as1->hrs_tot==2)
                                                {
                                                funcion_hace_nodo_y_copia(as1->hora);
                                                funcion_hace_nodo_y_copia(as1->hora+1);
                                                m1->asigna=1;
                                                if(clave_lab==1)
                                                        {
                                                        as4=as1;
                                                        for(as1=as;as1!=NULL;as1=as1->sig)
                                                                as1->lab=1;
                                                        as1=as4;
                                                        }
                                                clave_lab=0;
                                                diass++;
                                                if(diass>5)
                                                        diass=1;
                                                horass=7;
                                                }
                                        else
                                                {
                                                funcion_hace_nodo_y_copia(as1->hora);
                                                funcion_hace_nodo_y_copia(as1->hora+1);
                                                if(clave_lab==1)
                                                        {
                                                        as4=as1;
                                                        for(as1=as;as1!=NULL;as1=as1->sig)
                                                                as1->lab=1;
                                                        as1=as4;
                                                        }
                                                clave_lab=0;
                                                diass++;
                                                if(diass>5)
                                                        diass=1;
                                                horass=7;
                                                }
                                        }
                                else
                                        {funcion_horas();
                                        }
                                }
                        }
                else
                        {funcion_horas();
                        }
                }
else
        {if(as1->hora<21)
                {if(as1->hrs_tot==1)
                        {funcion_hace_nodo_y_copia(as1->hora);
                        m1->asigna=1;
                        if(clave_lab==1)
                                {
                                as4=as1;
                                for(as1=as;as1!=NULL;as1=as1->sig)
                                        as1->lab=1;
                                as1=as4;
                                }
                        clave_lab=0;
                        diass++;
                        if(diass>5)
                                diass=1;
                        horass=14;
```

```
                                        }
                                else
                                        {
                                        as1->hrs_tot=hortot;
                                        as1->hrs_prac=horprac;
                                        lista1(as1->hora+1);
                                        if(exist==0)
                                                {
                                                if(as1->hrs_tot==2)
                                                        {
                                                        funcion_hace_nodo_y_copia(as1->hora);
                                                        funcion_hace_nodo_y_copia(as1->hora+1);
                                                        m1->asigna=1;
                                                        if(clave_lab==1)
                                                                {
                                                                as4=as1;
                                                                for(as1=as;as1!=NULL;as1=as1->sig)
                                                                        as1->lab=1;
                                                                as1=as4;
                                                                }
                                                        clave_lab=0;
                                                        diass++;
                                                        if(diass>5)
                                                                diass=1;
                                                        horass=14;
                                                        }
                                                else
                                                        {
                                                        funcion_hace_nodo_y_copia(as1->hora);
                                                        funcion_hace_nodo_y_copia(as1->hora+1);
                                                        if(clave_lab==1)
                                                                {
                                                                as4=as1;
                                                                for(as1=as;as1!=NULL;as1=as1->sig)
                                                                        as1->lab=1;
                                                                as1=as4;
                                                                }
                                                        clave_lab=0;
                                                        diass++;
                                                        if(diass>5)
                                                                diass=1;
                                                        horass=14;
                                                        }
                                                }
                                        else
                                                {funcion_horas();
                                                }
                                        }
                                }
                        else
                                {funcion_horas();
                                }
                        }
                }
else
        {
        if((m1->hrs_teo+m1->hrs_prac)<=15)
                {if(as1->turno==0)
                        {if(as1->hora<12)
                                {if(as1->hrs_tot==1)
                                        {funcion_hace_nodo_y_copia(as1->hora);
                                        m1->asigna=1;
                                        if(clave_lab==1)
                                                {
                                                as4=as1;
                                                for(as1=as;as1!=NULL;as1=as1->sig)
                                                        as1->lab=1;
                                                as1=as4;
                                                }
                                        clave_lab=0;
                                        diass++;
                                        if(diass>5)
                                                diass=1;
                                        horass=7;
                                        }
                                else
                                        {
                                        as1->hrs_tot=hortot;
                                        as1->hrs_prac=horprac;
                                        lista1(as1->hora+1);
```

```
if(exist==0)
                {
                if(as1->hrs_tot==2)
                                {
                                funcion_hace_nodo_y_copia(as1->hora);
                                funcion_hace_nodo_y_copia(as1->hora+1);
                                m1->asigna=1;
                                if(clave_lab==1)
                                                {
                                                as4=as1;
                                                for(as1=as;as1!=NULL;as1=as1->sig)
                                                                as1->lab=1;
                                                as1=as4;
                                                }
                                clave_lab=0;
                                diass++;
                                if(diass>5)
                                                diass=1;
                                horass=7;
                                }
                else
                                {
                                as1->hrs_tot=hortot;
                                as1->hrs_prac=horprac;
                                lista1(as1->hora+2);
                                if(exist==0)
                                                {
                                                if(as1->hrs_tot==3)
                                                                {
                                                                funcion_hace_nodo_y_copia(as1->hora);
                                                                funcion_hace_nodo_y_copia(as1->hora+1);
                                                                funcion_hace_nodo_y_copia(as1->hora+2);
                                                                m1->asigna=1;
                                                                if(clave_lab==1)
                                                                                {
                                                                                as4=as1;
                                                                                for(as1=as;as1!=NULL;as1=as1->sig)
                                                                                                as1->lab=1;
                                                                                as1=as4;
                                                                                }
                                                                clave_lab=0;
                                                                diass++;
                                                                if(diass>5)
                                                                                diass=1;
                                                                horass=7;
                                                                }
                                                else
                                                                {
                                                                as1->hrs_tot=hortot;
                                                                as1->hrs_prac=horprac;
                                                                lista1(as1->hora+3);
                                                                if(exist==0)
                                                                                {
                                                                                if(as1->hrs_tot==4)
                                                                                                {
                                                                                                funcion_hace_nodo_y_copia(as1->hora);
                                                                                                funcion_hace_nodo_y_copia(as1->hora+1);
                                                                                                funcion_hace_nodo_y_copia(as1->hora+2);
                                                                                                funcion_hace_nodo_y_copia(as1->hora+3);
                                                                                                m1->asigna=1;
                                                                                                if(clave_lab==1)
                                                                                                {
                                                                                                as4=as1;
                                                                                                for(as1=as;as1!=NULL;as1=as1->sig)
```

```
                as1->lab=1;
as1=as4;
}
clave_lab=0;
diass++;
if(diass>5)
diass=1;
horass=7;
                                                                                                                }
                                                                                                        else
                                                                                                                {

funcion_hace_nodo_y_copia(as1->hora);
funcion_hace_nodo_y_copia(as1->hora+1);
funcion_hace_nodo_y_copia(as1->hora+2);
funcion_hace_nodo_y_copia(as1->hora+3);
as4=as1;
if(clave_lab==1)
{
as4=as1;
for(as1=as;as1!=NULL;as1=as1->sig)
                as1->lab=1;
as1=as4;
}
clave_lab=0;
diass++;
if(diass>5)
diass=1;
horass=7;
                                                                                                                }
                                                                                                }
                                                                                                        else
                                                                                                                funcion_horas();
                                                                                                        }
                                                                                                }
                                                                                        else
                                                                                                funcion_horas();
                                                                                        }
                                                                                }
                                                                        else
                                                                                {funcion_horas();
                                                                                }
                                                                        }
                                                                }
                                                        else
                                                                {funcion_horas();
                                                                }
                                                        }
                                                else
                                                        {if(as1->hora<19)
                                                                {if(as1->hrs_tot==1)
                                                                        {funcion_hace_nodo_y_copia(as1->hora);
                                                                        m1->asigna=1;
                                                                        if(clave_lab==1)
                                                                                {
                                                                                as4=as1;
```

```
                                                for(as1=as;as1!=NULL;as1=as1->sig)
                                                        as1->lab=1;
                                                as1=as4;
                                                }
                                        clave_lab=0;
                                        diass++;
                                        if(diass>5)
                                                diass=1;
                                        horass=14;
                                        }
                                else
                                        {
                                        as1->hrs_tot=hortot;
                                        as1->hrs_prac=horprac;
                                        lista1(as1->hora+1);
                                        if(exist==0)
                                                {
                                                if(as1->hrs_tot==2)
                                                        {funcion_hace_nodo_y_copia(as1->hora);
                                                        funcion_hace_nodo_y_copia(as1->hora+1);
                                                        m1->asigna=1;
                                                        if(clave_lab==1)
                                                                {
                                                                as4=as1;
                                                                for(as1=as;as1!=NULL;as1=as1->sig)
                                                                        as1->lab=1;
                                                                as1=as4;
                                                                }
                                                        clave_lab=0;
                                                        diass++;
                                                        if(diass>5)
                                                                diass=1;
                                                        horass=14;
                                                        }
                                                else
                                                        {
                                                        as1->hrs_tot=hortot;
                                                        as1->hrs_prac=horprac;
                                                        lista1(as1->hora+2);
                                                        if(exist==0)
                                                                {
                                                                if(as1->hrs_tot==3)
                                                                        {

        funcion_hace_nodo_y_copia(as1->hora);
                funcion_hace_nodo_y_copia(as1->hora+1);
                        funcion_hace_nodo_y_copia(as1->hora+2);
                                                                                m1->asigna=1;
                                                                                if(clave_lab==1)
                                                                                        {
                                                                                        as4=as1;

        for(as1=as;as1!=NULL;as1=as1->sig)
                                                                                                as1->lab=1;
                                                                                        as1=as4;
                                                                                        }
                                                                                clave_lab=0;
                                                                                diass++;
                                                                                if(diass>5)
                                                                                        diass=1;
                                                                                horass=14;
                                                                                }
                                                                        else
                                                                                {
                                                                                as1->hrs_tot=hortot;
                                                                                as1->hrs_prac=horprac;
                                                                                lista1(as1->hora+3);
                                                                                if(exist==0)
                                                                                        {
                                                                                        if(as1->hrs_tot==4)
                                                                                                {

        funcion_hace_nodo_y_copia(as1->hora);
                                funcion_hace_nodo_y_copia(as1->hora+1);
                        funcion_hace_nodo_y_copia(as1->hora+2);
                                funcion_hace_nodo_y_copia(as1->hora+3);
                                                                                                m1->asigna=1;
```

```
if(clave_lab==1)
{
as4=as1;
for(as1=as;as1!=NULL;as1=as1->sig)
        as1->lab=1;
as1=as4;
}
clave_lab=0;
diass++;
if(diass>5)
diass=1;
horass=14;
                                                                                    }
                                                                            else
                                                                                    {

funcion_hace_nodo_y_copia(as1->hora);
funcion_hace_nodo_y_copia(as1->hora+1);
funcion_hace_nodo_y_copia(as1->hora+2);
if(clave_lab==1)
{
as4=as1;
for(as1=as;as1!=NULL;as1=as1->sig)
        as1->lab=1;
as1=as4;
}
clave_lab=0;
diass++;
if(diass>5)
diass=1;
horass=14;
                                                                                    }
                                                                            }
                                                                    else
                                                                            funcion_horas();
                                                                    }
                                                            }
                                                    else
                                                            {funcion_horas();
                                                            }
                                                    }
                                            }
                                    else
                                            {funcion_horas();
                                            }
                                    }
                            }
                    else
                            {funcion_horas();
                            }
                    }
            else
                    {if((m1->hrs_teo+m1->hrs_prac)<=20)
```

```
{if(as1->turno==0)
        {if(as1->hora<11)
                {if(as1->hrs_tot==1)
                        {funcion_hace_nodo_y_copia(as1->hora);
                        m1->asigna=1;
                        if(clave_lab==1)
                                {
                                as4=as1;
                                for(as1=as;as1!=NULL;as1=as1->sig)
                                        as1->lab=1;
                                as1=as4;
                                }
                        clave_lab=0;
                        diass++;
                        if(diass>5)
                                diass=1;
                        horass=7;
                        }
                else
                        {
                        as1->hrs_tot=hortot;
                        as1->hrs_prac=horprac;
                        lista1(as1->hora+1);
                        if(exist==0)
                                {
                                if(as1->hrs_tot==2)
                                        {
                                        funcion_hace_nodo_y_copia(as1->hora);
                                        funcion_hace_nodo_y_copia(as1->hora+1);
                                        m1->asigna=1;
                                        if(clave_lab==1)
                                                {
                                                as4=as1;

for(as1=as;as1!=NULL;as1=as1->sig)
                                                                as1->lab=1;
                                                as1=as4;
                                                }
                                        clave_lab=0;
                                        diass++;
                                        if(diass>5)
                                                diass=1;
                                        horass=7;
                                        }
                                else
                                        {
                                        as1->hrs_tot=hortot;
                                        as1->hrs_prac=horprac;
                                        lista1(as1->hora+2);
                                        if(exist==0)
                                                {
                                                if(as1->hrs_tot==3)
                                                        {

funcion_hace_nodo_y_copia(as1->hora);

funcion_hace_nodo_y_copia(as1->hora+1);

funcion_hace_nodo_y_copia(as1->hora+2);
                                                                m1->asigna=1;
                                                                if(clave_lab==1)
                                                                        {


as4=as1;

for(as1=as;as1!=NULL;as1=as1->sig)

as1->lab=1;

as1=as4;
                                                                }
                                                                clave_lab=0;
                                                                diass++;
                                                                if(diass>5)


diass=1;
                                                                horass=7;
                                                                }
                                                        else
                                                                {
```

```
->hrs_tot=hortot;
->hrs_prac=horprac;
->hora+3);

        if(as1->hrs_tot==4)
        {
        funcion_hace_nodo_y_copia(as1->hora);
        funcion_hace_nodo_y_copia(as1->hora+1);
        funcion_hace_nodo_y_copia(as1->hora+2);
        funcion_hace_nodo_y_copia(as1->hora+3);
        m1->asigna=1;
        if(clave_lab==1)
                {
                as4=as1;
                for(as1=as;as1!=NULL;as1=as1->sig)
                        as1->lab=1;
                as1=as4;
                }
        clave_lab=0;
        diass++;
        if(diass>5)
                diass=1;
        horass=7;
        }

        {
        as1->hrs_tot=hortot;
        as1->hrs_prac=horprac;
        lista1(as1->hora+4);
        if(exist==0)
                {
                if(as1->hrs_tot==5)
                        {
                        funcion_hace_nodo_y_copia(as1->hora);
                        funcion_hace_nodo_y_copia(as1->hora+1);
                        funcion_hace_nodo_y_copia(as1->hora+2);
                        funcion_hace_nodo_y_copia(as1->hora+3);
                        funcion_hace_nodo_y_copia(as1->hora+4);
                        m1->asigna=1;
                        if(clave_lab==1)

as1-
as1-
lista1(as1-
if(exist==0)
                {

else
```

```
                        }
                        as4=as1;
                        for(as1=as;as1!=NULL;as1=as1->sig)
                                as1->lab=1;
                        as1=as4;
                        }
                clave_lab=0;
                diass++;
                if(diass>5)
                        diass=1;
                horass=7;
                }
        else
                {
                funcion_hace_nodo_y_copia(as1->hora);
                funcion_hace_nodo_y_copia(as1->hora+1);
                funcion_hace_nodo_y_copia(as1->hora+2);
                funcion_hace_nodo_y_copia(as1->hora+3);
                funcion_hace_nodo_y_copia(as1->hora+4);
                if(clave_lab==1)
                        {
                        as4=as1;
                        for(as1=as;as1!=NULL;as1=as1->sig)
                                as1->lab=1;
                        as1=as4;
                        }
                clave_lab=0;
                diass++;
                if(diass>5)
                        diass=1;
                horass=7;
                }
        }
else
        funcion_horas();
}
                                                                                }
                                                                        else
{funcion_horas();
                                                                                }
                                                                        }
                                                                }
                                                        else
```

```
                                                                                {funcion_horas();
                                                                                }
                                                                        }
                                                                }
                                                        else
                                                                {funcion_horas();
                                                                }
                                                        }
                                                }
                                        else
                                                {funcion_horas();
                                                }
                                        }
                                else
                                        {if(as1->hora<18)
                                                {if(as1->hrs_tot==1)
                                                        {funcion_hace_nodo_y_copia(as1->hora);
                                                        m1->asigna=1;
                                                        clave_lab=0;
                                                        diass++;
                                                        if(diass>5)
                                                                diass=1;
                                                        horass=14;
                                                        }
                                                else
                                                        {
                                                        as1->hrs_tot=hortot;
                                                        as1->hrs_prac=horprac;
                                                        lista1(as1->hora+1);
                                                        if(exist==0)
                                                                {
                                                                if(as1->hrs_tot==2)
                                                                        {
                                                                        funcion_hace_nodo_y_copia(as1->hora);
                                                                        funcion_hace_nodo_y_copia(as1->hora+1);
                                                                        m1->asigna=1;
                                                                        if(clave_lab==1)
                                                                                {
                                                                                as4=as1;

for(as1=as;as1!=NULL;as1=as1->sig)
                                                                                                as1->lab=1;
                                                                                as1=as4;
                                                                                }
                                                                        clave_lab=0;
                                                                        diass++;
                                                                        if(diass>5)
                                                                                diass=1;
                                                                        horass=14;
                                                                        }
                                                                else
                                                                        {
                                                                        as1->hrs_tot=hortot;
                                                                        as1->hrs_prac=horprac;
                                                                        lista1(as1->hora+2);
                                                                        if(exist==0)
                                                                                {
                                                                                if(as1->hrs_tot==3)
                                                                                        {

funcion_hace_nodo_y_copia(as1->hora);

funcion_hace_nodo_y_copia(as1->hora+1);

funcion_hace_nodo_y_copia(as1->hora+2);
                                                                                                        m1->asigna=1;
                                                                                                        if(clave_lab==1)
                                                                                                                {


as4=as1;

for(as1=as;as1!=NULL;as1=as1->sig)

as1->lab=1;

as1=as4;
                                                                                                                }
                                                                                                        clave_lab=0;
                                                                                                        diass++;
                                                                                                        if(diass>5)
```

```
        diass=1;

->hrs_tot=hortot;
->hrs_prac=horprac;
>hora+3);

        if(as1->hrs_tot==4)
        {
        funcion_hace_nodo_y_copia(as1->hora);
        funcion_hace_nodo_y_copia(as1->hora+1);
        funcion_hace_nodo_y_copia(as1->hora+2);
        funcion_hace_nodo_y_copia(as1->hora+3);
        m1->asigna=1;
        if(clave_lab==1)
                {
                as4=as1;
                for(as1=as;as1!=NULL;as1=as1->sig)
                        as1->lab=1;
                as1=as4;
                }
        clave_lab=0;
        diass++;
        if(diass>5)
                diass=1;
        horass=14;
        }
        {
        as1->hrs_tot=hortot;
        as1->hrs_prac=horprac;
        lista1(as1->hora+4);
        if(exist==0)
                {
                if(as1->hrs_tot==5)
                        {
                        funcion_hace_nodo_y_copia(as1->hora);
                        funcion_hace_nodo_y_copia(as1->hora+1);
                        funcion_hace_nodo_y_copia(as1->hora+2);
                        funcion_hace_nodo_y_copia(as1->hora+3);
                                                        horass=14;
                                                        }
                                                else
                                                        {
                                                        as1-
                                                        as1-
                                                        lista1(as1-
                                                        if(exist==0)
                                                                {
                                                                                        else
```

```
                funcion_hace_nodo_y_copia(as1->hora+4);
                m1->asigna=1;
                if(clave_lab==1)
                        {
                        as4=as1;
                        for(as1=as;as1!=NULL;as1=as1->sig)
                                as1->lab=1;
                        as1=as4;
                        }
                clave_lab=0;
                diass++;
                if(diass>5)
                        diass=1;
                horass=14;
                }
        else
                {
                funcion_hace_nodo_y_copia(as1->hora);
                funcion_hace_nodo_y_copia(as1->hora+1);
                funcion_hace_nodo_y_copia(as1->hora+2);
                funcion_hace_nodo_y_copia(as1->hora+3);
                funcion_hace_nodo_y_copia(as1->hora+4);
                if(clave_lab==1)
                        {
                        as4=as1;
                        for(as1=as;as1!=NULL;as1=as1->sig)
                        as1->lab=1;
                        as1=as4;
                        }
                clave_lab=0;
                diass++;
                if(diass>5)
                        diass=1;
                horass=14;
                }
        }
else
        funcion_horas();
}
                                                                        }
                                                                else
```

```
		{funcion_horas();
													}
											}
										}
								else
										{funcion_horas();
										}
								}
							}
						else
							{funcion_horas();
							}
						}
					}
				else
					{funcion_horas();
					}
				}
			}
		}
	}
else
	{funcion_horas();
	}
}

void copia(int hhora)
{
printf("°");
copiado=1;
para_variar=0;
cuennodos++;
cuennodosmateria++;
as3->materia=as1->materia;
as3->semestre=as1->semestre;
as3->turno=as1->turno;
as3->dia=as1->dia;
as3->hora=hhora;//as1->hora;
as3->grupo=as1->grupo;
if(clave_lab==1)
	{t0lab2--;}
else
	{
	if(as1->lab>0 && as1->hrs_prac>=1)
		asigna_lab(1);
	else
		as1->lab=0;
	}
as3->lab=as1->lab;
as3->carrera=as1->carrera;
for(i=0;i<=25;++i){as3->nom[i]=m1->nom[i];}
as3->area=m1->area;
as3->hrs_tot=as1->hrs_tot;
as3->hrs_prac=as1->hrs_prac;
if(as3->hrs_tot==1 && m1->hrs_prac!=0)
	{
	practics++;
	if(practics==tt1->mprac)

		{gruposs--;
		practics=0;
		for(m1=m;m1!=NULL;m1=m1->sig)
			m1->asigna=0;
		}
	}
if(tt1->grupos<1)
	{
	tt1->semestre=tt1->grupos=tt1->turno=0;
	tt1->carrera=tt1->hrs_tot=tt1->hrs_grupo=0;
	}
as1->hrs_tot--;
if(as1->hrs_prac>=1)
	as1->hrs_prac--;
//d1->horas++;
as3->sig=NULL;
ciclos=0;
}
```

```
void asigna_lab(int signal)
{
if(m1->hrs_prac!=0)
        {
        if(m1->turno==0)
                {
                if(t0lab1>0)
                        {
                        if(signal==0)
                                as1->lab=1;
                        if(signal==1)
                                t0lab1--;
                        }
                else
                        {
                        if(t0lab2>0)
                                {
                                if(signal==0)
                                        as1->lab=2;
                                if(signal==1)
                                        t0lab2--;
                                }
                        }
                }
        else
                {
                if(t1lab1>0)
                        {
                        if(signal==0)
                                as1->lab=1;
                        if(signal==1)
                                t1lab1--;
                        }
                else
                        {
                        if(t1lab2>0)
                                {
                                if(signal==0)
                                        as1->lab=2;
                                if(signal==1)
                                        t1lab2--;
                                }
                        }
                }
        }
else
        {
        as1->lab=0;
        }
}
```

/*---------- archivo mygraf.cpp ----------*/

Este archivo contiene funciones de gráficos que elaboran la pantalla de muestra de resultados, además del mouse. El resto de archivos contenidos en este apéndice contienen funciones de gráficos que permiten interactuar con los resultados mostrados en pantalla.

```
int size,ancho,alto;
int indic_sub=0,cuantos=0,x_sub,y_sub,x_h,y_h,x,xj,yy,y,xx;
int x_raton,y_raton;
char *opciones[4]={" Grupos "," Laboratorios "," Salir ",NULL};
char *laboratorios[5]={"Lab1 Matu","Lab2 Matu","Lab1 Vesp","Lab2 Vesp",NULL};
char *salidas[3]={"Imprimir"," Cerrar ",NULL};
void *saucer_h,*saucerop=NULL;

void boton(int xi,int yi,int color,int c_letra,char *tex,int tamano,int cuadro,int c_cuadro)
{
settextstyle(DEFAULT_FONT,HORIZ_DIR,tamano);
setfillstyle(SOLID_FILL,color);
ancho=textwidth(tex);
alto=textheight(tex);
bar(xi,yi,xi+ancho+10,yi+alto+10);
setcolor(c_letra);
line(xi,yi+alto+10,xi+ancho+10,yi+alto+10);
line(xi+ancho+10,yi,xi+ancho+10,yi+alto+10);
outtextxy(xi+5,yi+5,tex);
setcolor(c_cuadro);
```

```
if(cuadro==1)
                rectangle(xi,yi,xi+ancho+10,yi+alto+10);
return;
}

void linea(int num,int inicio1,int inicio2,int fin1,int fin2,int espacio,int color)
{
setcolor(color);
for(i=0;i<num;i++)
                {
                line(inicio1,inicio2,fin1,fin2);
                line(inicio1+1,inicio2,fin1+1,fin2);
                inicio1+=espacio;               fin1=inicio1;
                }
}


void blancos()
{
xi=50;
yi=150;
xf=580;
yf=360;

for(j=1,y=36;j<=9;j++)
                {
                boton(xi+10,yi+y,7,0,"          ",1,1,15);
                boton(xi+108,yi+y,15,0,"                                        ",1,1,0);
                y+=18;
                }
linea(4,xi+200,yi+36,xi+200,yi+200,80,0);
}

void menu(int x,int y,char *lista[])
{
for(i=0;lista[i]!=NULL;i++)
                {
                boton(x,y,7,11,lista[i],1,0,15);
                ancho=textwidth(lista[i]);
                x=x+ancho+5;
                ancho=textwidth(lista[i]);
                }
}

void sub_menu(int x,int y,char *lista[])
{
int size,ancho;
for(i=0,cuantos=0;lista[i]!=NULL;i++)
                {
                boton(x,y,8,15,lista[i],1,0,0);
                y=y+13;
                cuantos++;
                }
}

void g_portada()
{
int gd=DETECT,gm;
initgraph(&gd,&gm,"c:\\borlandc\\bgi");
cleardevice();
for(i=0;i<=30000;i++)
                {
                putpixel(random(640),random(480),LIGHTBLUE);
                }
rectangle(0,0,639,479);
}

void barra(int x,int y,char *lista[],int indic_sub)
{
int size,ancho,alto;
free(saucerop);
x_sub=x,y_sub=y;
ancho=textwidth(lista[0]);
alto=textheight(lista[0]);
size=imagesize(x,y,x+ancho+8,y+alto+6);
saucerop=malloc(size);
getimage(x,y,x+ancho+8,y+alto+6,saucerop);
setfillstyle(SOLID_FILL,WHITE);
bar(x,y,x+ancho+8,y+alto+6);
outtextxy(x+6,y+3,lista[indic_sub]);
```

```
}

void barra_h(int x,int y,char *lista)
{
int size,ancho,alto;
free(saucer_h);
x_h=x;
y_h=y;
ancho=textwidth(lista);
alto=textheight(lista);
size=imagesize(x,y,x+ancho+5,y+alto+6);
saucer_h=malloc(size);
getimage(x,y,x+ancho+5,y+alto+6,saucer_h);
setfillstyle(SOLID_FILL,WHITE);
bar(x,y,x+ancho+5,y+alto+6);
outtextxy(x+1,y+3,lista);
}

void grafico()
{
g_portada();
rectangle(1,1,638,478);
boton(200,15,7,1,"HORARIOS CONALEP",2,1,15);
settextstyle(DEFAULT_FONT,HORIZ_DIR,2);
setcolor(10);
outtextxy(208,23,"HORARIOS CONALEP");
menu(50,80,opciones);
xi=50;
yi=150;
xf=580;
yf=360;
x=107;
setfillstyle(SOLID_FILL,8);
bar(xi,yi,xi+530,yi+210);
boton(xi+10,yi+10,7,0," H O R A ",1,1,15);

boton(xi+x,yi+10,7,0," LUNES   MARTES  MIERCOLES JUEVES  VIERNES ",1,1,15);
linea(4,xi+200,yi+11,xi+200,yi+27,80,15);
blancos();
if(!mouse_reset())
        {
        printf(" ERROR: El mouse no se encuentra instalado");
        getch();
        exit(2);
        }
mouse_enable();
setfillstyle(1,7);
bar(2,469,637,477);
}

/*-------- archivo horass.cpp ----------------------------------------------------------------------------------------------*/
# include <c:\borlandc\bin\archivos.cpp>
void dias_a_colocar();
int xi,yi,xf,yf,turno;

void funcion_hora(int x,int y,int turno,char *lista)
{
setcolor(4);x=x+1;
if(turno==0)
        {
        if(as4->hora==7)
                {outtextxy(x,y,lista);}
        if(as4->hora==8)
                {outtextxy(x,y+17,lista);}
        if(as4->hora==9)
                {outtextxy(x,y+36,lista);}
        if(as4->hora==10)
                {outtextxy(x,y+53,lista);}
        if(as4->hora==11)
                {outtextxy(x,y+71,lista);}
        if(as4->hora==12)
                {outtextxy(x,y+90,lista);}
        if(as4->hora==13)
                {outtextxy(x,y+108,lista);}
        if(as4->hora==14)
                {outtextxy(x,y+125,lista);}
        }
if(turno==1)
        {
        if(as4->hora==14)
```

```
                        {outtextxy(x,y,lista);}
            if(as4->hora==15)
                        {outtextxy(x,y+17,lista);}
            if(as4->hora==16)
                        {outtextxy(x,y+36,lista);}
            if(as4->hora==17)
                        {outtextxy(x,y+53,lista);}
            if(as4->hora==18)
                        {outtextxy(x,y+71,lista);}
            if(as4->hora==19)
                        {outtextxy(x,y+90,lista);}
            if(as4->hora==20)
                        {outtextxy(x,y+108,lista);}
            if(as4->hora==21)
                        {outtextxy(x,y+125,lista);}
            }
setcolor(BLACK);
}

void horas(int turno)
{
setcolor(BLACK);
if(turno==1)
            {
            outtextxy(xi+15,yi+42,"14:00-15:00");
            outtextxy(xi+15,yi+59,"15:00-16:00");
            outtextxy(xi+15,yi+78,"16:00-17:00");
            outtextxy(xi+15,yi+95,"17:00-18:00");
            outtextxy(xi+15,yi+113,"18:00-19:00");
            outtextxy(xi+15,yi+132,"19:00-20:00");
            outtextxy(xi+15,yi+150,"20:00-21:00");
            outtextxy(xi+15,yi+167,"21:00-22:00");
            }
else
            {
            outtextxy(xi+15,yi+42," 7:00-8:00");
            outtextxy(xi+15,yi+59," 8:00-9:00");
            outtextxy(xi+15,yi+78," 9:00-10:00");
            outtextxy(xi+15,yi+95,"10:00-11:00");
            outtextxy(xi+15,yi+113,"11:00-12:00");
            outtextxy(xi+15,yi+132,"12:00-13:00");
            outtextxy(xi+15,yi+150,"13:00-14:00");
            outtextxy(xi+15,yi+167,"14:00-15:00");
            }
}

void dias_a_colocar(int turno,char *lista)
{
if(as4->dia==1)
            {
            funcion_hora(xi+110,yi+42,turno,lista);
            }
if(as4->dia==2)
            {
            funcion_hora(xi+200,yi+42,turno,lista);
            }
if(as4->dia==3)
            {
            funcion_hora(xi+280,yi+42,turno,lista);
            }
if(as4->dia==4)
            {
            funcion_hora(xi+360,yi+42,turno,lista);
            }
if(as4->dia==5)
            {
            funcion_hora(xi+440,yi+42,turno,lista);
            }
}


void imprime_grupo()
{
exist=0;
for(as4=as2;as4!=NULL;as4=as4->sig)
            {
            if(as4->turno==turno && as4->semestre==semestre
                        && as4->carrera==carrera && as4->grupo==grupo)
                        {
                        if(exist==0)
```

```
                                        {
                                        as3=as4;
                                        horas(turno);
                                        exist++;
                                        }
                                dias_a_colocar(turno,as4->nom);
                                as5=as4->sig;
                                }
                        }
as4=as2;
}

void imprime_lab(int lab,int turno)
{
int num_de_grupo=0;
exist=0;
for(as4=as2;as4!=NULL;as4=as4->sig)
                        {
                        if(as4->turno==turno && as4->lab==lab)
                                {
                                if(exist==0)
                                        {
                                        horas(turno);exist++;
                                        }
                                num_de_grupo=0;
                                num_de_grupo=as4->semestre*1000+num_de_grupo;
                                num_de_grupo=as4->turno*100+num_de_grupo+100;
                                num_de_grupo=as4->carrera*10+num_de_grupo;
                                num_de_grupo=as4->grupo+num_de_grupo;
                                dias_a_colocar(turno,fcvt(num_de_grupo,0,0,0));
                                }
                        }
free(as4);
as4=NULL;
}


/*------- archivo flechas.cpp ----------------------------------------------------------------------------------------------------*/
int f_der_izq(int indic,int x,int y,int z,char *listai,char *lista[])
{
barra_h(x,y,listai);
sub_menu(x,z,lista);
barra(x,z,lista,0);
indic_sub=0;
return(indic);
}

int flecha_abajo(int indic,int indic_sub,int sube_baja)
{
putimage(x_sub,y_sub,saucerop,0);
free(saucerop);
saucerop=NULL;
alto=13;
if(indic==1)
                {
                if(sube_baja==1)
                        barra(x_sub,y_sub+alto,grupos,indic_sub);
                if(sube_baja==0)
                        barra(x_sub,y_sub-alto,grupos,indic_sub);
                }
if(indic==2)
                {
                if(sube_baja==1)
                        barra(x_sub,y_sub+alto,laboratorios,indic_sub);
                if(sube_baja==0)
                        barra(x_sub,y_sub-alto,laboratorios,indic_sub);
                }
if(indic==3)
                {
                if(sube_baja==1)
                        barra(x_sub,y_sub+alto,salidas,indic_sub);
                if(sube_baja==0)
                        barra(x_sub,y_sub-alto,salidas,indic_sub);
                }
return(indic_sub);
}


/*------- archivo move_bar.cpp----------------------------------------------------------------------------------------------------*/
void apuntanodo(int dia,int hora1,int hora2);
```

```
void valordia(int);
int modifica=0,senalado=0,dias,hecho=0;

int mueve_barra(int x,int y,char *lista[],int indic_sub)
{
mouse_disable();
putimage(x_sub,y_sub,saucerop,0);
free(saucerop);
saucerop=NULL;
indic_sub=indic_sub;
barra(x,y,lista,indic_sub);
mouse_enable();
return(indic_sub);
}

char f_clikk(char opc)
{
int color;
for(;;)
        {
        mouse_read_cursor();
        if(opc=='g' && mouse_status==0)
                {
                if(mouse_x>=50 && mouse_x<=87)
                        {if(mouse_y>=100 && mouse_y<=114)
                                {
                                if(indic_sub!=0)
                                        {
                                        indic_sub=mueve_barra(50,100,grupos,0);
                                        }
                                }
                        if(mouse_y>=115 && mouse_y<=126)
                                {
                                if(indic_sub!=1)
                                        {
                                        indic_sub=mueve_barra(50,113,grupos,1);
                                        }
                                }
                        if(mouse_y>=127 && mouse_y<=139)
                                {
                                if(indic_sub!=2)
                                        {
                                        indic_sub=mueve_barra(50,126,grupos,2);
                                        }
                                }
                        if(mouse_y>=140 && mouse_y<=152)
                                {
                                if(indic_sub!=3)
                                        {
                                        indic_sub=mueve_barra(50,139,grupos,3);
                                        }
                                }
                        if(mouse_y>=153 && mouse_y<=163)
                                {
                                if(indic_sub!=4)
                                        {


                                        indic_sub=mueve_barra(50,152,grupos,4);
                                        }
                                }
                        if(mouse_y>=164 && mouse_y<=176)
                                {
                                if(indic_sub!=5)
                                        {
                                        indic_sub=mueve_barra(50,165,grupos,5);
                                        }
                                }
                        if(mouse_y>=177 && mouse_y<=189)
                                {
                                if(indic_sub!=6)
                                        {
                                        indic_sub=mueve_barra(50,178,grupos,6);
                                        }
                                }
                        if(mouse_y>=190 && mouse_y<=202)
                                {
                                if(indic_sub!=7)
                                        {
                                        indic_sub=mueve_barra(50,191,grupos,7);
```

```
                                        }
                                }
                        if(mouse_y>=203 && mouse_y<=215)
                                {
                                if(indic_sub!=8)
                                        {
                                        indic_sub=mueve_barra(50,204,grupos,8);
                                        }
                                }
                        if(mouse_y>=216 && mouse_y<=228)
                                {
                                if(indic_sub!=9)
                                        {
                                        indic_sub=mueve_barra(50,217,grupos,9);
                                        }
                                }
                        if(mouse_y>=229 && mouse_y<=241)
                                {
                                if(indic_sub!=10)
                                        {
                                        indic_sub=mueve_barra(50,230,grupos,10);
                                        }
                                }
                        if(mouse_y>=242 && mouse_y<=254)
                                {
                                if(indic_sub!=11)
                                        {
                                        indic_sub=mueve_barra(50,243,grupos,11);
                                        }
                                }
                        if(mouse_y>=255 && mouse_y<=267)
                                {
                                if(indic_sub!=12)
                                        {
                                        indic_sub=mueve_barra(50,256,grupos,12);
                                        }
                                }
                        if(mouse_y>=268 && mouse_y<=280)
                                {
                                if(indic_sub!=13)
                                        {
                                        indic_sub=mueve_barra(50,269,grupos,13);
                                        }
                                }
                        if(mouse_y>=281 && mouse_y<=293)
                                {
                                if(indic_sub!=14)
                                        {
indic_sub=mueve_barra(50,282,grupos,14);
                                        }
                                }
                        if(mouse_y>=294 && mouse_y<=306)
                                {
                                if(indic_sub!=15)
                                        {
                                        indic_sub=mueve_barra(50,293,grupos,15);
                                        }
                                }
                        if(mouse_y>=307 && mouse_y<=319)
                                {
                                if(indic_sub!=16)
                                        {
                                        indic_sub=mueve_barra(50,308,grupos,16);
                                        }
                                }
                        if(mouse_y>=320 && mouse_y<=332)
                                {
                                if(indic_sub!=17)
                                        {
                                        indic_sub=mueve_barra(50,321,grupos,17);
                                        }
                                }
                        if(mouse_y>=333 && mouse_y<=345)
                                {
                                if(indic_sub!=18)
                                        {
                                        indic_sub=mueve_barra(50,334,grupos,18);
                                        }
                                }
```

```
                        if(mouse_y>=346 && mouse_y<=358)
                                {
                                if(indic_sub!=19)
                                        {
                                        indic_sub=mueve_barra(50,347,grupos,19);
                                        }
                                }

                        }
                }
if(opc=='l' && mouse_status==0)
                {
                if(mouse_x>=125 && mouse_x<=202)
                        {if(mouse_y>=100 && mouse_y<=114)
                                {
                                if(indic_sub!=0)
                                        {
                                        indic_sub=mueve_barra(125,100,laboratorios,0);
                                        }
                                }
                        if(mouse_y>=115 && mouse_y<=126)
                                {
                                if(indic_sub!=1)
                                        {
                                        indic_sub=mueve_barra(125,113,laboratorios,1);
                                        }
                                }
                        if(mouse_y>=127 && mouse_y<=139)
                                {
                                if(indic_sub!=2)
                                        {
                                        indic_sub=mueve_barra(125,126,laboratorios,2);
                                        }
                                }
                        if(mouse_y>=140 && mouse_y<=152)
                                {
                                if(indic_sub!=3)
                                        {
                                        indic_sub=mueve_barra(125,139,laboratorios,3);
                                        }
                                }
                        }
                }
if(opc=='s' && mouse_status==0)
                {
                if(mouse_x>=240 && mouse_x<=301)
                        {if(mouse_y>=100 && mouse_y<=114)
                                {
                                if(indic_sub!=0)
                                        {
                                        indic_sub=mueve_barra(240,100,salidas,0);
                                        }
                                }
                        if(mouse_y>=115 && mouse_y<=126)
                                {
                                if(indic_sub!=1)
                                        {
                                        indic_sub=mueve_barra(240,113,salidas,1);
                                        }
                                }
                        }
                }

if(mouse_status==1)
                {
                if(mouse_y>=82 && mouse_y<97)
                        {
                        if(mouse_x>=50 && mouse_x<120)
                                {opc='g';indic_sub=0;break;}
                        if(mouse_x>=125 && mouse_x<243)
                                {opc='l';break;}
                        if(mouse_x>=240 && mouse_x<302)
                                {opc='s';break;}
                        }
                if(mouse_x>=50 && mouse_x<=87 && opc=='g')
                        {if(mouse_y>=100 && mouse_y<=114)
                                {opc=13;indic=1;indic_sub=0;break;}
                        if(mouse_y>=113 && mouse_y<=127)
                                {opc=13;indic=1;indic_sub=1;break;}
                        if(mouse_y>=126 && mouse_y<=140)
```

```
                {opc=13;indic=1;indic_sub=2;break;}
        if(mouse_y>=139 && mouse_y<=153)
                {opc=13;indic=1;indic_sub=3;break;}
        if(mouse_y>=152 && mouse_y<=166)
                {opc=13;indic=1;indic_sub=4;break;}
        if(mouse_y>=165 && mouse_y<=179)
                {opc=13;indic=1;indic_sub=5;break;}
        if(mouse_y>=178 && mouse_y<=192)
                {opc=13;indic=1;indic_sub=6;break;}
        if(mouse_y>=191 && mouse_y<=205)
                {opc=13;indic=1;indic_sub=7;break;}
        if(mouse_y>=204 && mouse_y<=218)
                {opc=13;indic=1;indic_sub=8;break;}
        if(mouse_y>=217 && mouse_y<=231)
                {opc=13;indic=1;indic_sub=9;break;}
        if(mouse_y>=230 && mouse_y<=242)
                {opc=13;indic=1;indic_sub=10;break;}
        if(mouse_y>=243 && mouse_y<=255)
                {opc=13;indic=1;indic_sub=11;break;}
        if(mouse_y>=256 && mouse_y<=268)
                {opc=13;indic=1;indic_sub=12;break;}
        if(mouse_y>=269 && mouse_y<=281)
                {opc=13;indic=1;indic_sub=13;break;}
        if(mouse_y>=282 && mouse_y<=294)
                {opc=13;indic=1;indic_sub=14;break;}
        if(mouse_y>=295 && mouse_y<=307)
                {opc=13;indic=1;indic_sub=15;break;}
        if(mouse_y>=308 && mouse_y<=320)
                {opc=13;indic=1;indic_sub=16;break;}
        if(mouse_y>=321 && mouse_y<=333)
                {opc=13;indic=1;indic_sub=17;break;}
        if(mouse_y>=334 && mouse_y<=346)
                {opc=13;indic=1;indic_sub=18;break;}
        if(mouse_y>=347 && mouse_y<=359)
                {opc=13;indic=1;indic_sub=19;break;}
        if(mouse_y>=360 && mouse_y<=372)
                {opc=13;indic=1;indic_sub=20;break;}
        }
if(mouse_x>=158 && mouse_x<=567)
        {if(mouse_y>=186 && mouse_y<=350)
                {
                if(modifica==1)
                        {
                        for(;;)
                                {
                                mouse_read_cursor();
                                if(mouse_status==1)
                                        {
                                        if(mouse_x>=158 && mouse_x<=250)
                                                {
                                                valordia(1);
                                                }
                                        if(mouse_x>=251 && mouse_x<=330)
                                                {
                                                valordia(2);
                                                }
                                        if(mouse_x>=331 && mouse_x<=410)
                                                {
                                                valordia(3);
                                                }
                                        if(mouse_x>=411 && mouse_x<=490)
                                                {
                                                valordia(4);
                                                }
                                        if(mouse_x>=491 && mouse_x<=567)
                                                {
                                                valordia(5);
                                                }
                                        }
                                if(mouse_status==2)
                                        {
                                        modifica=0;
                                        color=getcolor();
                                        setcolor(7);
                                        outtextxy(5,470,"Modificacion activa");
                                        setcolor(color);
                                        break;
                                        }
                                }
                        }
```

```
                                        }
                                }
                        if(mouse_x>=125 && mouse_x<=202 && opc=='l')
                                {if(mouse_y>=100 && mouse_y<=114)
                                        {opc=13;indic=2;indic_sub=0;break;}
                                if(mouse_y>=113 && mouse_y<=127)
                                        {opc=13;indic=2;indic_sub=1;break;}
                                if(mouse_y>=126 && mouse_y<=140)
                                        {opc=13;indic=2;indic_sub=2;break;}
                                if(mouse_y>=139 && mouse_y<=153)
                                        {opc=13;indic=2;indic_sub=3;break;}
                                }
                        if(mouse_x>=240 && mouse_x<301 && opc=='s')
                                {if(mouse_y>=100 && mouse_y<=114)
                                        {opc=13;indic=3;indic_sub=0;break;}
                                if(mouse_y>=113 && mouse_y<=127)
                                        {opc=13;indic=3;indic_sub=1;break;}
                                }

                        }
                if(mouse_status==2)
                        {
                        opc=27;break;
                        }
                }
return(opc);
}

/*---------- archivo opciones.cpp ----------*/
# include <c:\borlandc\bin\imprime.cpp>
void *sauarrayy=NULL;
int dato=0,color;
div_t xdato;

void flechas_y_letras()
{
                        if(opc==77) //derecha
                                {
                                indic_sub=0;
                                if(indic>0 && indic<3)
                                        {
                                        if(indic==1)
                                                {
                                                putimage(50,80,sauarrayy,0);
                                                free(sauarrayy);
                                                indic=f_der_izq(indic,125,82,100,opciones[1],laboratorios);
                                                indic++;
                                                }
                                        else
                                                {
                                                if(indic==2)
                                                        {
                                                        putimage(125,80,sauarrayy,0);
                                                        indic=f_der_izq(indic,240,82,100,opciones[2],salidas);
                                                        indic++;
                                                        }
                                                }
                                        }
                                if(dato!=0)
                                        horas(turno);
                                }
                        if(opc==75) //izq
                                {
                                indic_sub=0;
                                if(indic>0 && indic<=3)
                                        {
                                        if(indic==2)
                                                {
                                                if(dato!=0)
                                                        horas(turno);
                                                putimage(125,80,sauarrayy,0);
                                                free(sauarrayy);
                                                indic=f_der_izq(indic,50,82,100,opciones[0],grupos);
                                                indic--;
                                                }
                                        else
                                                {
                                                if(indic==3)
                                                        {
                                                        putimage(240,80,sauarrayy,0);
```

```
                                        free(sauarrayy);
                                        indic=f_der_izq(indic,125,82,100,opciones[1],laboratorios);
                                        indic--;
                                        }
                                }
                        }
                }
if(opc==80)//abajo
                {
                if(indic>0 && indic<=3)
                        {
                        if(indic_sub>=0 && indic_sub<=cuantos-2)
                                {
                                indic_sub++;
                                indic_sub=flecha_abajo(indic,indic_sub,1);
                                }
                        }
                }
if(opc==72)//arriba
                {
                if(indic>0 && indic<=3)
                        {
                        if(indic_sub>=1)
                                {
                                indic_sub--;
                                indic_sub=flecha_abajo(indic,indic_sub,0);
                                }
                        }
                }
if(opc=='G' || opc=='g')
                {
                color=getcolor();
                setcolor(7);
                outtextxy(5,470,"Modificacion activa");
                setcolor(color);

                indic_sub=0;
                if(indic==2)
                        {
                        if(sauarrayy!=NULL)
                                {putimage(125,80,sauarrayy,0);
                                free(sauarrayy);}
                        }
                if(indic==3)
                        {
                        if(sauarrayy!=NULL)
                                {putimage(240,80,sauarrayy,0);
                                free(sauarrayy);}
                        }
                if(indic!=1)
                        {
                        ancho=textwidth(grupos[0]);
                        size=imagesize(50,80,50+ancho+50,400);
                        sauarrayy=malloc(size);
                        getimage(50,80,50+ancho+50,400,sauarrayy);

                        if(dato!=0)
                                horas(turno);
                        indic=f_der_izq(indic,50,82,100,opciones[0],grupos);
                        indic=1;
                        }
                }
if(opc=='L' || opc=='l')
                {
                color=getcolor();
                setcolor(7);
                outtextxy(5,470,"Modificacion activa");
                setcolor(color);
                indic_sub=0;
                if(indic==1)
                        {

                        if(sauarrayy!=NULL)
                                {putimage(50,80,sauarrayy,0);
                                free(sauarrayy);}
                        }
                if(indic==3)
                        {
                        if(sauarrayy!=NULL)
                                {putimage(240,80,sauarrayy,0);
```

```
                                        free(sauarrayy);}
                                }
                        if(indic!=2)
                                {
                                ancho=textwidth(laboratorios[0]);
                                size=imagesize(125,80,125+ancho-50,160);
                                sauarrayy=malloc(size);
                                getimage(125,80,125+ancho+50,160,sauarrayy);

                                indic=f_der_izq(indic,125,82,100,opciones[1],laboratorios);
                                indic=2;
                                }
                        if(dato!=0)
                                horas(turno);
                        }
                if(opc=='S'||opc=='s')
                        {
                        color=getcolor();
                        setcolor(7);
                        outtextxy(5,470,"Modificacion activa");
                        setcolor(color);
                        indic_sub=0;
                        if(indic==1)
                                {
                                if(sauarrayy!=NULL)
                                        {putimage(50,80,sauarrayy,0);
                                        free(sauarrayy);}
                                }
                        if(indic==2)
                                {
                                if(sauarrayy!=NULL)
                                        {putimage(125,80,sauarrayy,0);
                                        free(sauarrayy);}
                                }
                        if(indic!=3)
                                {
                                ancho=textwidth(salidas[0]);
                                size=imagesize(240,80,240+ancho+20,135);
                                sauarrayy=malloc(size);
                                getimage(240,80,240+ancho+20,135,sauarrayy);
                                indic=f_der_izq(indic,240,82,100,opciones[2],salidas);
                                indic=3;
                                }
                        if(dato!=0)
                                horas(turno);
                        }
}

void opcion()
{
for(;;)
        {
        opc=f_clikk(opc);
        mouse_disable();
                if(opc==27)//escape
                        {
                        if(indic==1)
                                {
                                putimage(50,80,sauarrayy,0);
                                free(sauarrayy);
                                if(dato!=0)
                                        horas(turno);
                                }
                        if(indic==2)
                                {putimage(125,80,sauarrayy,0);
                                free(sauarrayy);}
                        if(indic==3)
                                {putimage(240,80,sauarrayy,0);
                                free(sauarrayy);}
                        indic=0;
                        }
                if(opc==13)//enter
                        {
                        if(indic==1)
                                {
                                putimage(50,80,sauarrayy,0);
                                free(sauarrayy);
                                blancos();
                                for(i=0;i<indic_sub;++i);
```

```
                dato=atoi(grupos[i]);
                setcolor(15);
                setfillstyle(SOLID_FILL,8);
                bar(440,95,580,135);
                outtextxy(450,110,"GRUPO:");
                boton(500,100,15,9,grupos[i],2,1,7);
                settextstyle(0,0,1);
                xdato=div(dato,10);
                grupo=xdato.rem;
                xdato=div(xdato.quot,10);
                carrera=xdato.rem;
                xdato=div(xdato.quot,10);
                semestre=xdato.quot;
                turno=xdato.rem-1;
                imprime_grupo();
                indic=0;
                modifica=1;
                color=getcolor();
                setcolor(BLACK);
                outtextxy(5,470,"Modificacion activa");
                setcolor(color);
                }
if(indic==2)
                {
                putimage(125,80,sauarrayy,0);
                free(sauarrayy);
                blancos();
                setcolor(15);
                setfillstyle(SOLID_FILL,8);
                bar(440,95,580,135);
                outtextxy(445,105,"LABORATORIO:");
                outtextxy(445,120,"TURNO:");
                settextstyle(0,0,1);
                if(indic_sub==0)
                                {
                                imprime_lab(1,0);
                                turno=0;
                                setcolor(10);
                                outtextxy(550,105,"1");
                                outtextxy(500,120,"MATUTINO");
                                setcolor(0);
                                }
                if(indic_sub==1)
                                {
                                imprime_lab(2,0);
                                turno=0;
                                setcolor(10);
                                outtextxy(550,105,"2");
                                outtextxy(500,120,"MATUTINO");
                                setcolor(0);
                                }
                if(indic_sub==2)
                                {
                                imprime_lab(1,1);
                                turno=1;
                                setcolor(10);
                                outtextxy(550,105,"1");
                                outtextxy(500,120,"VESPERTINO");
                                setcolor(0);
                                }
                if(indic_sub==3)
                                {
                                imprime_lab(2,1);
                                turno=1;
                                setcolor(10);
                                outtextxy(550,105,"2");
                                outtextxy(500,120,"VESPERTINO");
                                setcolor(0);
                                }
                indic=0;
                modifica=0;
                }
if(indic==3)
                {
                if(indic_sub==0)
                                {
                                impresora();
                                opc='s';
                                }
                if(indic_sub==1)
```

```
                                                {
                                                free(as3);
                                                closegraph();
                                                exit(0);
                                                }
                                        }
                                }
                flechas_y_letras();
                mouse_enable();
                }
}

/*---------- archivo modifica.cpp ----------*/
void valordia(int dia)
{
if(mouse_y>=186 && mouse_y<=204)
                {
                apuntanodo(dia,7,14);
                }
if(mouse_y>=205 && mouse_y<=222)
                {
                apuntanodo(dia,8,15);
                }
if(mouse_y>=223 && mouse_y<=240)
                {
                apuntanodo(dia,9,16);
                }
if(mouse_y>=241 && mouse_y<=258)
                {
                apuntanodo(dia,10,17);
                }
if(mouse_y>=259 && mouse_y<=276)
                {
                apuntanodo(dia,11,18);
                }
if(mouse_y>=277 && mouse_y<=294)
                {
                apuntanodo(dia,12,19);
                }
if(mouse_y>=295 && mouse_y<=312)
                {
                apuntanodo(dia,13,20);
                }
if(mouse_y>=313 && mouse_y<=330)
                {
                apuntanodo(dia,14,21);
                }
}

void c_lista(struct asignaciones *ptr1,struct asignaciones *ptr2)
{
exist=0;
for(as4=as2;as4!=NULL;)
                {
                if(as4->turno==ptr1->turno)
                                {
                                if(as4->lab==ptr1->lab)
                                                {
                                                if(as4->dia==ptr2->dia)
                                                                {
                                                                if(as4->hora!=ptr2->hora)
                                                                                {
                                                                                exist=0;
                                                                                as4=as4->sig;
                                                                                }
                                                                else
                                                                                {
                                                                                exist=1;
                                                                                as4=NULL;
                                                                                }
                                                                }
                                                else
                                                                as4=as4->sig;
                                                }
                                else
                                                as4=as4->sig;
                                }
                else
                                as4=as4->sig;
                }
```

```
}

void cambio(struct asignaciones *ptr1,struct asignaciones *ptr2)
{
int dia,hora;
dia=ptr2->dia;
hora=ptr2->hora;
ptr2->dia=ptr1->dia;
ptr2->hora=ptr1->hora;
ptr1->dia=dia;
ptr1->hora=hora;
}

void sonido(int frec)
{
sound(frec);
delay(30);
nosound();
}

void imprime(int soundd)
{
if(soundd)
        sonido(2500);
senalado=0;
mouse_disable();
blancos();
imprime_grupo();
mouse_enable();
}

void mensaje()
        {
        sonido(500);
        color=getcolor();
        setcolor(7);
        outtextxy(5,470,"Modificacion activa");
        setcolor(BLACK);
        outtextxy(5,470,"Posicion no valida...");
        delay(1200);
        setcolor(7);
        outtextxy(5,470,"Posicion no valida...");
        setcolor(BLACK);
        outtextxy(5,470,"Modificacion activa");
        imprime(0);
        setcolor(color);
        }


void apuntanodo(int dia,int hora1,int hora2)
{
int hora,color;
exist=0;
if(as3->turno==0)
        {
        ptr1=as3;
        while(ptr1!=as5 && (ptr1->dia!=dia || ptr1->hora!=hora1))
                {
                hora=hora1;
                ptr1=ptr1->sig;
                }
        }
else
        {
        ptr1=as3;
        while(ptr1!=as5 && (ptr1->dia!=dia || ptr1->hora!=hora2))
                {
                hora=hora2;
                ptr1=ptr1->sig;
                }
        }
if(ptr1!=as5)
        {
        if(senalado==0)
                {
                setcolor(LIGHTRED);
                outtextxy(mouse_x-6,mouse_y-6,"*");
                setcolor(BLACK);
                ptr2=ptr1;
```

```
                senalado++;
                dias=dia;
                }
        else
                {
                if(ptr1!=ptr2)
                        {
                        setcolor(LIGHTRED);
                        outtextxy(mouse_x-6,mouse_y-6,"*");
                        setcolor(BLACK);
                        senalado=0;
                        if(ptr2->lab==ptr1->lab)
                                {
                                cambio(ptr2,ptr1);
                                imprime(1);
                                }
                        else
                                {
                                if(ptr2->lab!=0 && ptr1->lab==0)
                                        {
                                        c_lista(ptr2,ptr1);
                                        if(exist==0)
                                                {
                                                cambio(ptr2,ptr1);
                                                imprime(1);
                                                }
                                        else
                                                mensaje();
                                        }
                                else
                                        {
                                        if(ptr2->lab==0 && ptr1->lab!=0)
                                                {
                                                c_lista(ptr1,ptr2);
                                                if(exist==0)
                                                        {
                                                        cambio(ptr2,ptr1);
                                                        imprime(1);
                                                        }
                                                else
                                                        mensaje();
                                                }
                                        else
                                                {
                                                c_lista(ptr1,ptr2);
                                                if(exist==0)
                                                        {
                                                        c_lista(ptr2,ptr1);
                                                        if(exist==0)
                                                                {
                                                                cambio(ptr2,ptr1);
                                                                imprime(1);
                                                                }
                                                        else
                                                                mensaje();
                                                        }
                                                else
                                                        mensaje();
                                                }
                                        }
                                }
                        }
                }
        }
else
        {
        if(senalado!=0)
                {
                if(ptr2->lab==0)
                        {
                        ptr2->dia=dia;
                        ptr2->hora=hora;
                        imprime(1);
                        }
                else
                        {
                        ptr1=NULL;
                        ptr1->dia=dia;
                        ptr1->hora=hora;
                        c_lista(ptr2,ptr1);
```

```
                                        if(exist==0)
                                                {
                                                cambio(ptr2,ptr1);
                                                imprime(1);
                                                }
                                        else
                                                mensaje();
                                        }
                                }
                        }
}

/*-------- archivo imprime.cpp ----------------------------------------------------------------------------------------------------*/
#include <c:\borlandc\bin\pantalla.cpp>
#define VOLCADO "\33E"

impresora()
{
if((psalida=fopen("PRN","wb"))==NULL)
                {
                puts("\nimpresora no preparada\n");
                getch();
                exit(0);
                }
pantalla();
fprintf(psalida,VOLCADO);
fclose(pentrada);
fclose(psalida);
return(0);
}


/*-------- archivo pantalla.cpp ----------------------------------------------------------------------------------------------------*/
FILE *pentrada,*psalida;

char *busca(int dia, int hora)
{
for(as4=as3;as4!=as5;)
                {
                if(as4->dia==dia && as4->hora==hora)
                                {break;}
                else
                                {as4=as4->sig;}
                }
if(as4!=as5)
                {
                while(strlen(as4->nom)<10)
                                {strcat(as4->nom," ");}
                return(as4->nom);
                }
else
                {
                return("         ");
                }
}

void tabla(int turno,char *vargrupo)
{
int varhora;
if(turno==0)
                varhora=7;
else
                varhora=14;

fprintf(psalida,"\Tabla de horario del grupo: %s\n",vargrupo);
fprintf(psalida,"\n\r³ H o r a ³ L u n e s ³ M a r t e s ³ Miercoles ³ J u e v e s ³ Viernes ³");
if(varhora==7)
                {
                fprintf(psalida,"\n\r³%d:00-%d:00  ³%s ³%s ³%s ³%s
³%s³",varhora,varhora+1,busca(1,varhora),busca(2,varhora),busca(3,varhora),busca(4,varhora),busca(5,varhora));
                fprintf(psalida,"\n\r³%d:00-%d:00  ³%s ³%s ³%s ³%s
³%s³",varhora+1,varhora+2,busca(1,varhora+1),busca(2,varhora+1),busca(3,varhora+1),busca(4,varhora+1),busca(5,varhora+1));
                fprintf(psalida,"\n\r³%d:00-%d:00  ³%s ³%s ³%s ³%s
³%s³",varhora+2,varhora+3,busca(1,varhora+2),busca(2,varhora+2),busca(3,varhora+2),busca(4,varhora+2),busca(5,varhora+2));
                }
else
                {
                fprintf(psalida,"\n\r³%d:00-%d:00 ³%s ³%s ³%s ³%s
³%s³",varhora,varhora+1,busca(1,varhora),busca(2,varhora),busca(3,varhora),busca(4,varhora),busca(5,varhora));
```

```
                fprintf(psalida,"\n\r%d:00-%d:00 %s  %s  %s  %s
%s",varhora+1,varhora+2,busca(1,varhora+1),busca(2,varhora+1),busca(3,varhora+1),busca(4,varhora+1),busca(5,varhora+1));
                fprintf(psalida,"\n\r%d:00-%d:00 %s  %s  %s  %s
%s",varhora+2,varhora+3,busca(1,varhora+2),busca(2,varhora+2),busca(3,varhora+2),busca(4,varhora+2),busca(5,varhora+2));
                }
fprintf(psalida,"\n\r%d:00-%d:00 %s  %s  %s  %s
%s",varhora+3,varhora+4,busca(1,varhora+3),busca(2,varhora+3),busca(3,varhora+3),busca(4,varhora+3),busca(5,varhora+3));
fprintf(psalida,"\n\r%d:00-%d:00 %s  %s  %s  %s
%s",varhora+4,varhora+5,busca(1,varhora+4),busca(2,varhora+4),busca(3,varhora+4),busca(4,varhora+4),busca(5,varhora+4));
fprintf(psalida,"\n\r%d:00-%d:00 %s  %s  %s  %s
%s",varhora+5,varhora+6,busca(1,varhora+5),busca(2,varhora+5),busca(3,varhora+5),busca(4,varhora+5),busca(5,varhora+5));
fprintf(psalida,"\n\r%d:00-%d:00 %s  %s  %s  %s
%s",varhora+6,varhora+7,busca(1,varhora+6),busca(2,varhora+6),busca(3,varhora+6),busca(4,varhora+6),busca(5,varhora+6));
fprintf(psalida,"\n\r%d:00-%d:00 %s  %s  %s  %s
%s",varhora+7,varhora+8,busca(1,varhora+7),busca(2,varhora+7),busca(3,varhora+7),busca(4,varhora+7),busca(5,varhora+7));
}


void pantalla()
{
tabla(0,"5105");
}


/*-------- archivo mouse.cpp ----------------------------------------------------------------------------------------*/
#ifndef __DOS_H
#include <dos.h>
#endif
# include <dos.h>
int mouse_status;
int mouse_x;
int mouse_y;


int  mouse_reset(void);
void mouse_enable(void);
void mouse_disable(void);
void mouse_read_cursor(void);
void mouse_horizontal_range(int xmin, int xmax);
void mouse_vertical_range(int ymin, int ymax);
void mouse_set(int, int, int, int);

int mouse_reset(void)
{
                _AX=0;
  geninterrupt(0x33);
  return(_AX);
}               /* mouse_reset */


void mouse_enable(void)
{
  _AX=1;
  geninterrupt(0x33);
}               /* mouse_enable */


void mouse_disable(void)
{
  _AX=2;
  geninterrupt(0x33);
}               /* mouse_disable */


void mouse_read_cursor(void)
{
  _AX=3;
  geninterrupt(0x33);
  mouse_status=_BX;
  mouse_x=_CX;
  mouse_y=_DX;
}               /* mouse_read_cursor */


void mouse_horizontal_range(int xmin, int xmax)
{
  _AX=7;
  _CX=xmin;
  _DX=xmax;
  geninterrupt(0x33);
```

```
}               /* mouse_vertical_range */

void mouse_vertical_range(int ymin, int ymax)
{
  _AX=8;
  _CX=ymin;
  _DX=ymax;


  geninterrupt(0x33);
}               /* mouse_vertical_range */

void mouse_set(int a, int b, int c, int d)
{
  mouse_horizontal_range(a, b);
  mouse_vertical_range(c, d);
}               /* Mouse_set */
```

conalep

PLANTEL " GRAL. ANTONIO DE LEÓN " CLAVE:145

HORARIOS DE GRUPO

GRUPO: 3101 SEMESTRE: SEPTIEMBRE 99-FEBRERO 00

CARRERA: PROFESIONAL TECNICO EN ADMINISTRACION CLAVE: ADMO 6217

HORARIO	LUNES	MARTES	MIÉRCOLES	JUEVES	VIERNES
7-8	ADMON. DE LA PROD.	ADMON. DE LA PROD.	ADMON DE ALM. E INV.		COMUNICACIÓN DE CIENCIA Y TEC.
8-9	ADMON. DE LA PROD.	ADMON. DE LA PROD.	ADMON DE ALM. E INV.	ADMON DE ALM. E INV.	COMUNICACIÓN DE CIENCIA Y TEC.
9-10	COMUNICACIÓN DE CIENCIA Y TEC.	ADMON DE ALM. E INV.		CONTROL DE ACT. FIJOS	CONTROL DE ACTIVOS FIJOS
10-11	MECANICA Y CALOR	ADMON DE ALM. E INV.	HISTORIA DE MEXICO	CONTROL DE ACT. FIJOS	CONTROL DE ACTIVOS FIJOS
11-12	MECANICA Y CALOR	HISTORIA DE MEXICO	HISTORIA DE MEXICO		ADMON. DE LA PRODUCCION
12-13		MATEMATICAS TECNICAS	CALIDAD TOTAL		MATEMATICAS TECNICAS
13-14	MECANICA Y CALOR	MATEMATICAS TECNICAS	CALIDAD TOTAL		MATEMATICAS TECNICAS
14-15		CALIDAD TOTAL	MATEMATICAS TECNICAS		

APÉNDICE 2

conalep

PLANTEL " GRAL. ANTONIO DE LEÓN " CLAVE:145

HORARIOS DE GRUPO

GRUPO: 3202 SEMESTRE: SEPTIEMBRE 99-FEBRERO 00

CARRERA: PROFESIONAL TECNICO EN CONTABILIDAD FINANCIERA Y FISCAL

HORARIO	LUNES	MARTES	MIÉRCOLES	JUEVES	VIERNES
14-15	REGLAMENTACION TRIBUTARIA	HISTORIA DE MEXICO	REGLAMENTACION TRIBUTARIA	GESTION MERCANTIL	GESTION MERCANTIL
15-16	MATEMATICAS TECNICAS	MATEMATICAS TECNICAS	MATEMATICAS TECNICAS	MATEMATICAS TECNICAS	MATEMATICAS TECNICAS
16-17	CALIDAD TOTAL	CALIDAD TOTAL	CALIDAD TOTAL	COMUNICACIÓN DE CIENCIA Y TEC.	COMUNICACIÓN DE CIENCIA Y TEC.
17-18	OPERACIÓN CONTABLE DE SOC.	MECANICA Y CALOR	MECANICA Y CALOR		COMUNICACIÓN DE CIENCIA Y TEC.
18-19	OPERACIÓN CONTABLE DE SOC.	HISTORIA DE MEXICO	MECANICA Y CALOR	REGLAMENTACION TRIBUTARIA	OPERACIÓN CONTABLE DE SOC.
19-20	OPERACIÓN CONTABLE DE SOC.	HISTORIA DE MEXICO	GESTION MERCANTIL	REGLAMENTACION TRIBUTARIA	HISTORIA DE MEXICO
20-21	OPERACIÓN CONTABLE DE SOC.	HISTORIA DE MEXICO	GESTION MERCANTIL	REGLAMENTACION TRIBUTARIA	HISTORIA DE MEXICO

conalep

PLANTEL " GRAL. ANTONIO DE LEÓN " CLAVE:145

HORARIOS DE GRUPO

GRUPO: 3103 SEMESTRE: SEPTIEMBRE 99-FEBRERO 00

CARRERA: PROFESIONAL TECNICO EN CONTABILIDAD FINANCIERA Y FISCAL CLAVE: CONT6237

HORARIO	LUNES	MARTES	MIÉRCOLES	JUEVES	VIERNES
7-8	COMUNICACION		OPERACIÓN CONTABLE DE SOC.		REGLAMENTACION TRIBUTARIA
8-9	COMUNICACION		OPERACIÓN CONTABLE DE SOC.	OPERACIÓN CONTABLE DE SOC.	REGLAMENTACION TRIBUTARIA
9-10	REGLAMENTACION TRIBUTARIA	HISTORIA DE MEXICO	HISTORIA DE MEXICO	OPERACIÓN CONTABLE DE SOC.	REGLAMENTACION TRIBUTARIA
10-11	REGLAMENTACION TRIBUTARIA	HISTORIA DE MEXICO	COMUNICACIÓN DE CIENCIA Y TEC.	OPERACIÓN CONTABLE DE SOC.	GESTION MARCANTIL
11-12		GESTION MARCANTIL	MATEMATICAS TECNICAS		GESTION MERCANTIL
12-13	MATEMATICAS	GESTION MERCANTIL	MATEMATICAS TECNICAS	MATEMATICAS TECNICAS	MATEMATICAS TECNICAS
13-14	CALIDAD TOTAL	CALIDAD TOTAL	MECANICA Y CALOR	MECANICA Y CALOR	MECANICA Y CALOR
14-15	CALIDAD TOTAL				

conalep

PLANTEL " GRAL. ANTONIO DE LEÓN " CLAVE:145

121

HORARIOS DE GRUPO

GRUPO: 3204 SEMESTRE: SEPTIEMBRE 99-FEBRERO 00

CARRERA: PROFESIONAL TECNICO EN INFORMATICA CLAVE: INFO7247

HORARIO	LUNES	MARTES	MIÉRCOLES	JUEVES	VIERNES
14-15		HISTORIA DE MEXICO			MECANICA Y CALOR
15-16	CALIDAD TOTAL	CALIDAD TOTAL	CALIDAD TOTAL	COMUNICACIÓN DE C. Y TEC.	MECANICA Y CALOR
16-17	MATEMATICAS TECNICAS	MATEMATICAS TECNICAS	MATEMATICAS TECNICAS	MATEMATICAS TECNICAS	MATEMATICAS TECNICAS
17-18	COMUNICACIÓN DE C. Y TEC.	INST. BAJO AMB. GRAF. Y APLIC. C.C.1	ADMON. DE SIST. DE COMPUTO C.C.1	PROG. EST. EN PSEUDOCODIGO	PROG. EST. EN PSEUDOCODIGO C.C.2
18-19	COMUNICACIÓN DE C. Y TEC.	INST. BAJO AMB. GRAF. Y APLIC. C.C.1	ADMON. DE SIST. DE COMPUTO C.C.1	INST. BAJO AMB. GRAF. Y APLIC.	PROG. EST. EN PSEUDOCODIGO C.C.2
19-20	HISTORIA DE MEXICO	PROG. EST. EN PSEUDOCODIGO C.C.1	MATEMATICAS DISCRETAS C.C.2	MECANICA Y CALOR	MATEMATICAS DISCRETAS C.C.1
20-21	HISTORIA DE MEXICO	ADMON. DE SIST. DE COMPUTO	MATEMATICAS DISCRETAS		MATEMATICAS DISCRETAS C.C.1

HORARIOS DE GRUPO

GRUPO: 3205 SEMESTRE: SEPTIEMBRE 99-FEBRERO 00

CARRERA: PROFESIONAL TECNICO EN INFORMATICA

HORARIO	LUNES	MARTES	MIÉRCOLES	JUEVES	VIERNES
14-15	INSTALACION DE AMBIENTE C.C.2	MECANICA Y CALOR	INSTALACION DE AMBIENTE C.C.2	CALIDAD TOTAL	PROG. EST. EN PSEUDOC. C.C.1
15-16	INSTALACION DE AMBIENTE	HISTORIA DE MEXICO		CALIDAD TOTAL	MATEMATICAS DISCRETAS C.C.2
16-17	ADMON. DE SIST. DE COM.	MECANICA Y CALOR	MECANICA Y CALOR	CALIDAD TOTAL	MATEMATICAS DISCRETAS C.C.2
17-18	MATEMATICAS TECNICAS	MATEMATICAS TECNICAS	MATEMATICAS TECNICAS	MATEMATICAS TECNICAS	MATEMATICAS TECNICAS
18-19	COMUNICACIÓN DE CIENCIA Y TEC.	ADMON. DE SIST. DE COM. C.C.2	HISTORIA DE MEXICO	COMUNICACIÓN DE CIENCIA Y TEC.	PROG. EST. EN PSEUDOC.
19-20	MATEMATICAS DISCRETAS	ADMON. DE SIST. DE COM. C.C.2	HISTORIA DE MEXICO	COMUNICACIÓN DE CIENCIA Y TEC.	PROG. EST. EN PSEUDOC. C.C.2
20-21	MATEMATICAS DISCRETAS				PROG. EST. EN PSEUDOC. C.C.2

APÉNDICE 3

MANUAL DEL USUARIO

A continuación se describe el funcionamiento del programa realizado para la asignación de horarios de clase del CONALEP y la forma en que puede interactuar con el programa.

Instalación del programa

Para instalar el programa en su computadora siga estos pasos.

1. Inserte el disco que contiene el programa a instalar en su unidad de disco flexible correspondiente (A ó B).

2. Seleccione la unidad en la que se insertó el disco. Si utiliza el sistema operativo MS-DOS deberá teclear el siguiente comando ***A:*** y presionar la tecla ***enter***. Si utiliza el sistema operativo windows, diríjase al explorador de windows y seleccione la unidad correspondiente al disco.

3. Escriba la instrucción ***ins_prog*** y presione la tecla ***enter***. El programa ***ins_prog*** se encarga de copiar los archivos necesarios en la unidad de disco duro en la carpeta de nombre ***horarios***.

4. Una vez finalizada la copia de archivos, estos se localizan en la ruta ***c: \horarios *** y está listo para ejecutarse.

Ejecución del programa

Para la ejecución del programa siga los siguientes pasos.

1. Ejecute el sistema operativo MS-DOS.

2. Localice y seleccione la carpeta ***horarios***. Al realizar este paso, en el indicador de MS-DOS aparecerá la siguiente línea:

   C: \horarios

3. Escriba la siguiente palabra ***comm*** y pulse la tecla ***enter***. Este es el nombre del programa de asignación de horarios, al escribir este nombre en el indicador del sistema operativo iniciará su ejecución.

4. Al iniciar su ejecución, el programa le muestra el siguiente mensaje:
   " Teclee el número que corresponde a la actual distribución de horarios...
   1. ***Septiembre – Febrero***
   2. ***Marzo - Agosto***"

Usted sólo podrá teclear el número 1 o el número 2, cualquier otro valor no será tomado en cuenta por el sistema y nuevamente le pedirá su respuesta.

5. Al teclear su respuesta, el programa le muestra el siguiente mensaje: ***"Tamaño de la población a generar (mayor que 0)"***, en este momento el programa le pide que teclee un valor mayor que 0, este valor servirá al programa para aplicar la técnica de algoritmos genéticos y realizar la asignación de horarios. El valor que sea tecleado deberá estar en un rango de 1 a 200, cualquier otra respuesta no será válida y el programa pedirá nuevamente su respuesta.

6. Con estos datos de entrada, el programa carga en la memoria de la computadora las tablas de datos que se van a distribuir, si alguna tabla no se encuentra, el programa enviará un mensaje de error.

7. El programa inicia la distribución de materias, y el avance en esta tarea se verá en pantalla por una serie de barras, este proceso continua hasta que aparece la siguiente figura A1.

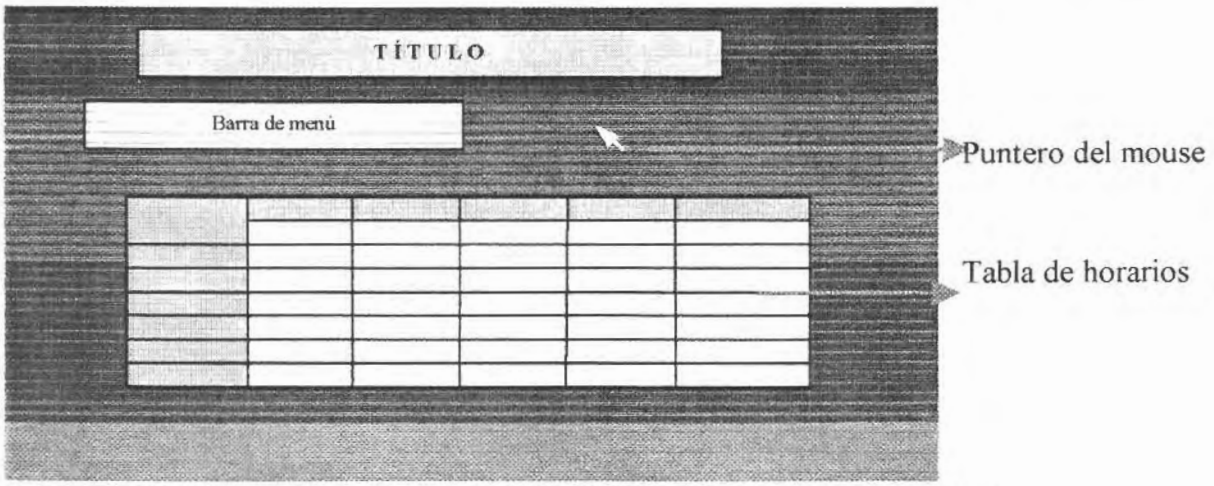


Figura. A1. *Presentación de resultados con gráficos y mouse*

8. La gráfica en la pantalla de la computadora indica que el programa ha realizado la asignación de horarios y ahora esta disponible para consultar las asignaciones realizadas.

9. El programa le muestra una barra de menú con las siguientes opciones:

   Grupos Laboratorios Profesores Salir

   En este menú se puede seleccionar cualquier opción utilizando el apuntador del mouse, posicionándose en el nombre de la opción que se desea ejecutar y presionando el botón derecho del mouse se desplegará el contenido de cada opción.

10. La opción ***Grupos*** presentará una lista con los números de cada grupo formado y a los que se le han asignado horarios de clase, la opción ***Laboratorios*** contiene la lista de laboratorios a los que se les han asignado horarios de clase, la opción ***Profesores*** presentará la lista por áreas y nombres de cada profesor, al seleccionar cualquiera de estas opciones aparecerá en la tabla mostrada en pantalla a asignación que el programa realizó.

11. En la tabla mostrada por cada grupo seleccionada, tendrá la opción de modificar las asignaciones de cada materia, utilizando el mouse y marcando las materias que desea modificar, en pantalla aparecerá un asterisco en la materia que desea modificar, si la modificación es válida, el programa hace el cambio de asignación y actualiza las tablas que se hayan modificado, si no es válida, entonces envía un mensaje de asignación no disponible.
    Si desea salir de la modificación de asignaciones, bastará con presionar el botón derecho del mouse y de esta forma se podrá seleccionar alguna otra opción del menú.

12. Para salir del sistema, elija la opción ***salir*** del menú principal y elija la opción ***Cerrar***. En esta opción ***Salir*** se muestra la opción ***Imprimir***, esta opción le permitirá imprimir la tabla que en ese momento se está consultando.

Cabe aclarar que el programa sólo reconoce el funcionamiento del mouse para seleccionar o moverse a través de la tabla y no reconocerá ninguna tecla que se presione.