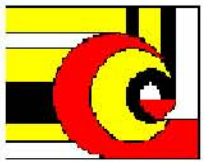
INSTITUTO POLITÉCNICO NACIONAL
CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN
MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN



BÚSQUEDA TABÚ PARA RESOLVER UN PROBLEMA DE CAMBIOS DE ESCUELA EN EDUCACIÓN BÁSICA DE MÉXICO DISTRITO FEDERAL.

TESIS QUE PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA: JORGE SÁNCHEZ HERRERA
DIRECTOR DE TESIS
DR. LUIS BERNARDO MORALES MENDOZA
CODIRECTOR DE TESIS
DR. CARLOS AGUILAR IBÁÑEZ

México D. F., 2003

ÍNDICE DE ALTO NIVEL

ÍNDICE DETALLADO

RELACIÓN DE ILUSTRACIONES

ESTRUCTURA GENERAL

En esta tesis se presenta la siguiente información:

Resumen en español, véase a continuación.

Resumen en ingles, véase a continuación.

Objetivos, véase sección 1.2.

Problemática, véase sección 2.1.

Formulación del problema, véase sección 2.3.

Problemas combinatorios, véase sección 3.1.

Resultados globales, véase sección 6.2.

Conclusiones, véase sección 6.3.

Resumen

Los métodos heurísticos son poderosas herramientas para abordar problemas cuyas soluciones de cómputo demandan grandes costos o que son computacionalmente intratables.

El método heurístico "Búsqueda Tabú", permite encontrar buenas soluciones a problemas de optimización combinatoria.

En esta tesis, utilizando Búsqueda Tabú, se presenta una solución al problema de cambios de escuela de alumnos que no están de acuerdo con la secundaria asignada en la ciudad de México Distrito Federal. Este problema se modela como un sistema de programación entera binaria, donde a partir de 12,561 solicitudes de alumnos y 806 escuelas participantes, se da lugar a un gran sistema de optimización combinatoria de 12,561 variables binarias y 779 restricciones.

Abstract

The heuristics methods are powerful tools to solve problems whose solutions require a lot of compute costs or are non-treatable in a computational way.

The Tabu Search method achieves to find good solutions to combinatory optimization problems.

In this thesis, Tabu Search method is used to solve the problem of changes of students, which do not agree with the assigned school in Mexico City. This problem is modeled as a binary integer programming system. This has to solve 12,561 school requests and 806 schools. With this information we build a binary integer programming system with 12,561 variables and 779 constrains.

CAPÍTULO 1.
INTRODUCCIÓN

En este capítulo se presenta el estado del arte del uso del Meta-heurístico Búsqueda Tabú en la solución de problemas de naturaleza combinatoria.

1.1 ESTADO DEL ARTE DE BÚSQUEDA TABÚ

La dificultad y abundancia de problemas de optimización encontrados en los campos de las telecomunicaciones, logística, planeación financiera, transporte y producción, han motivado el desarrollo de poderosas técnicas de optimización. Estas técnicas son usualmente el resultado de adaptar ideas de una variedad de áreas de investigación con la esperanza de encontrar procedimientos eficientes y manejables en el tratamiento de problemas de optimización; así, encontramos procedimientos como el Recocido Simulado RS, que está basado en analogías de procesos físicos metalúrgicos; Algoritmos Genéticos AGs, que buscan imitar el fenómeno biológico de la reproducción evolutiva y, Búsqueda Tabú BT, basado en derivar y explotar un conjunto de principios de resolución inteligente de problemas.

El método BT, está basado en procedimientos para cruzar límites de viabilidad u óptimos locales normalmente tratados como barreras, estos procedimientos buscan sistemáticamente imponer y liberar restricciones para la diversificación de las exploraciones. La exploración se basa en la selección de una solución inicial no siempre factible, que a partir de una operación llamada movimiento, se desplaza a soluciones adyacentes contenidas en subespacios de solución llamados vecindarios. Una de las características principales del método, es que en la búsqueda de la solución óptima, cada movimiento que se va realizando pasa a ser temporalmente prohibido (cuando un movimiento es prohibido se dice que es tabú), lo que evita quedar atrapado en ciclos u óptimos locales. Los criterios de terminación son dados a través de un determinado número de iteraciones o el alcanzar un valor de aspiración.

La moderna forma de Búsqueda Tabú deriva de Glover [1], contribuciones adicionales fueron dadas a partir del uso de alternativas para mejorar la exploración y obtener soluciones más satisfactorias; posteriormente muchos investigadores han hecho

importantes aportaciones a BT, lo que ha contribuido a la evolución del método y el crecimiento del uso de éste en la solución de problemas de optimización combinatoria.

Una de las primeras aplicaciones de Búsqueda Tabú en planeación fue realizada por Widmer and Hertz [2], para la solución del problema de un taller que tiene que producir un conjunto de productos. Se supone por ejemplo, el caso de un taller que puede realizar un solo producto a la vez (orden *i*), el que se tiene contratado a entregar en g_i días, a partir de una cierta fecha base, y que además tiene una duración de trabajo de d_i ($d_i > 0$) días y el cual se asocia una multa de p_i pesos por día de retraso después de los g_i días estipulados. Se supone que el taller recibe *n* órdenes diferentes de trabajo en la fecha base. ¿Cuál debe ser el orden de secuenciación de trabajos que minimice el costo penal total?. Sea por ejemplo el siguiente caso:

Orden número (i)	Día de entrega contratado a partir del día de hoy (g_i)	Duración del trabajo en días (d_i)	Multa por día de retraso a partir de la fecha de entrega en pesos por día (p_i)
1	2	5	$5/día
2	4	4	$4/día
3	8	3	$2/día
4	12	5	$1/día
5	13	2	$7/día
6	17	7	$1/día

Ilustración 1. Datos del problema de asignación de trabajo en el taller [1].

Widmer and Hertz, utilizaron un procedimiento que considera vecindarios definidos por movimientos de intercambio. En cada iteración el mejor movimiento no tabú es ejecutado, a la vez que es evaluada la función objetivo *c(x)*. El tiempo tabú es exclusivamente puesto a un valor de 7 movimientos y las restricciones tabú se basan sobre los atributos emparejados (índices de tareas y posición). Los criterios de terminación están especificados como un máximo número de iteraciones.

Los reportes realizados por investigadores y usuarios de BT, abarcan problemas de diversos campos, siendo las áreas más representativas aún cuando no limitativas: la planeación, la programación de producción, las redes de telecomunicaciones, los problemas del agente viajero, los problemas tipo mochila, el diseño de redes, la coloración de grafos, el problema de la ruta de un vehículo, la planeación de proyectos con recursos restringidos, los sistemas de bases de datos, la optimización global, entre otros.

Los laboratorios de investigación de Mitsubishi Electric [3], están trabajando en un proyecto denominado Human-Guided Tabu Search, donde reportan que han extendido los trabajos pasados en el ciclo de optimización, permitiendo que usuarios humanos guíen un poderoso algoritmo de búsqueda llamado "Búsqueda Tabú". Han Encontrado dos ventajas de esta integración en el proceso de optimización. Primera, el humano puede guiar sistemas para satisfacer soluciones con varias restricciones. Segundo, los usuarios humanos pueden guiar su destreza visual, habilidad para aprender y estrategias sensibles para mejorar el rendimiento del cómputo del algoritmo de búsqueda.

J.E. Beasley, H. Howells y J. Sonander [4], presentaron en febrero del 2001, un reporte donde mencionan haber encontrado mejoras a la alerta de conflictos aéreos utilizando Búsqueda Tabú, este trabajo reportado fue llevado a cabo en la National Air Traffic Services en el Reino Unido. Consiste en una herramienta de optimización para auxiliar la mejora de la eficacia de uno de sus sistemas de seguridad de tráfico aéreo. Este sistema es computarizado y está continuamente monitoreando los datos del radar y alerta a los controladores del tráfico aéreo si es que detecta una situación donde dos aeronaves están en peligro de encontrarse una con otra. Junto con el programa que actualiza el sistema hay una larga lista de parámetros. La elección apropiada de estos parámetros es una tarea que se realizaba manualmente, la aportación que los autores

reportan consiste en utilizar Búsqueda Tabú para la elección automática de los parámetros.

Jiefeng Xu, Steve Y. Chiu and Fred Glover, publicaron en la revista Management Science [5], el uso de Búsqueda Tabú para optimizar una red de telecomunicaciones privada en topología de anillo, donde mencionan que uno de los problemas en el diseño de redes privadas en la industria de las telecomunicaciones, es la interconexión de un conjunto de clientes localizados a través de un anillo de oficinas. Lo que se busca, es minimizar el costo de la tarifa y proveer fiabilidad. El Método Tabú utilizado, incorpora conceptos de memoria larga, selección de movimientos aleatorios, evaluación de jerarquía de movimientos, lista de estrategias candidatas y una elite de soluciones recobradas estratégicamente; los resultados obtenidos con las pruebas de datos muestran que el método empleado encuentra resultados óptimos para todos los casos mostrados. Aunque estos problemas pueden ser resueltos con métodos exactos usando el algoritmo "branch and cut", el método Búsqueda Tabú encuentra el optimo en una tercera parte del tiempo empleado por el método exacto.

Se prevé que los principios del meta-heurístico Búsqueda Tabú serán cada vez más utilizados y referenciados en la solución de problemas de optimización. Prueba de ello es el trabajo "Reactive Local Search", realizado por R. Battiti [6]. El autor propone que la definición de la lista tabú, sea aprendida de manera automática por la reacción a la ocurrencia de ciclos; si en la búsqueda aparece un excesivo número de soluciones frecuentes, entonces la búsqueda es diversificada haciendo un número de movimientos aleatorios proporcional al promedio de movimientos de la longitud del ciclo.

1.2 OBJETIVOS

Generar un modelo de Programación Entera Binaria que represente adecuadamente el problema de cambios de escuelas para la obtención de resultados óptimos.

Diseñar y Construir una aplicación informática basada en el Meta-Heurístico "Búsqueda Tabú", para proponer mejores soluciones en el proceso de cambios de escuela.

Contribuir con el Sistema Educativo Mexicano, a través de la aportación de mejores soluciones, en la atención a la demanda ciudadana.

CAPÍTULO 2
DESCRIPCIÓN DEL PROBLEMA

En este capítulo se describe el problema de cambios de escuelas de educación secundaria y se realiza el planteamiento matemático.

2.1 PROBLEMÁTICA

La Secretaría de Educación Pública SEP en la Ciudad de México Distrito Federal, con la finalidad de garantizar la atención escolar al 100 % de los aspirantes a ingresar a secundaria y, proporcionar el mayor grado de satisfacción en cuanto a las escuelas solicitadas por la población, cada año realiza un proceso de asignación de los alumnos que ingresan a primer grado en escuelas secundarias oficiales (federales). Una vez publicados los resultados de asignación, aproximadamente 14,000 de un total de 140,000 niños, se muestran inconformes con la escuela y turno de asignación. Para atender dicha inconformidad la SEP ha instrumentado un proceso de cambios de escuela que consiste en:

a) Durante un periodo de cinco días hábiles a partir de la entrega de resultados, los aspirantes inconformes, acuden a la escuela de su preferencia a llenar una solicitud de cambio de escuela, en dicha solicitud expresan el motivo por el cual desean esa asignación.

b) La SEP concentra las solicitudes y a partir del motivo expresado se le asigna una prioridad de cambio a cada solicitud.

c) A través de una aplicación informática, se realiza el proceso de cambios y se informa a la población los resultados del proceso.

La problemática es enumerada de la siguiente forma:

1.- Atender una situación de índole social bajo las premisas de brindar la máxima satisfacción a la población demandante de los servicios de educación secundaria y reducir la posibilidad de deserción escolar o rezagos educativos, generados a partir de

que no sean compatibles la escuela y el horario asignados con aquellos que se ajustan a las necesidades de los alumnos de cercanía de centros escolares y turno.

2.-Atender criterios de racionalidad, definidos a partir de la capacidad de grupos de alumnos, considerando el número de aulas disponibles en cada escuela y el respeto a la norma en cuanto a la relación de alumnos por grupo (a lo más 50 alumnos por grupo para el caso en cuestión).

3.-Se considera que la solución actual, corresponde a un óptimo local y que es factible de mejorarse, sin embargo a la fecha no ha sido posible.

4.-Una de las limitantes que el proceso actual tiene es que no proporciona una solución al caso o casos donde grupos de alumnos se encuentran en lo que se conoce como abrazos ejemplo:

Caso 1:		Caso *n*:
$A_{i,k}$ solicita E_j ; $A_{j,k+1}$ solicita E_i ; $A_{j+1,k+3}$ solicita E_m; $A_{i,k+2}$ solicita E_m;; $A_{i,k}$, es el alumno *k* que está asignado en la escuela *i*. E_i es la escuela *i.* En ambas E_j,E_i no existen lugares disponibles, mientras que en E_m existe un lugar disponible.	...	Uno de los casos críticos que se puede presentar es: $A_{1,1}$ solicita E_2, $A_{2,2}$ solicita E_3, ..., $A_{n-1,m}$ solicita E_n; $A_{n,m+1}$ solicita E_1 y $A_{n+1,m+2}$ solicita E_1; salvo en la E_1 que existe un lugar, en las restantes E_i no existen lugares disponibles.

$$i \neq j \neq k \neq m \neq n$$
$$i, j, k, m, n > 0$$

Ilustración 2. Casos de abrazos circulares en las solicitudes de cambios de escuela

Para ilustrar la problemática partamos del ejemplo, donde se tienen 7 solicitudes de cambio con lugares disponibles sólo en una escuela.

a) Datos de alumnos

No. Consecutivo de alumno	Escuela de origen	Escuela solicitada	Calificación	Peso
1	2	1	22	1
2	1	3	32	1
3	4	5	36	1
4	1	2	40	1
5	3	5	22	1
6	1	2	40	1
7	6	5	31	1

Ilustración 3. Ejemplificación de la problemática, alumnos participantes

b) Datos de escuelas

No. de escuela	Lugares disponibles
1	0
2	0
3	0
4	0
5	1
6	0

Ilustración 4. Ejemplificación de la problemática, escuelas participantes

Solución proporcionada por el Sistema que se encuentra en operación actualmente:

Estado Inicial

Alumnos

No. Consecutivo de alumno	Escuela de origen	Escuela solicitada	Calificación	Prioridad	Alumno asignado
4	1	2	40	1	0
6	1	2	40	1	0
3	4	5	36	1	0
2	1	3	32	1	0
7	6	5	31	1	0
1	2	1	22	1	0
5	3	5	22	1	0

Escuelas

No. De escuela	Lugares disponibles
1	0
2	0
3	0
4	0
5	1
6	0

Ilustración 5. Ejemplificación de la problemática

1.- La tabla de alumnos se ordena de manera descendente por los campos prioridad y calificación.

2.- La tabla de alumnos Se recorre secuencialmente de inicio a fin; para cada registro (alumno), se verifica la disponibilidad de lugares en la escuela solicitada, en caso de que exista al menos un lugar, se procede a realizar el cambio de escuela, se incrementa en uno los lugares disponibles de la escuela origen se decrementa en uno, los lugares disponibles en la escuela destino.

Estado Final

Alumnos

No. Consecutivo de alumno	Escuela de origen	Escuela solicitada	Calificación	Prioridad	Alumno asignado
4	1	2	40	1	0
6	1	2	40	1	0
3	4	5	36	1	1
2	1	3	32	1	0
7	6	5	31	1	0
1	2	1	22	1	0
5	3	5	22	1	0

Escuelas

No. de escuela	Lugares disponibles
1	0
2	0
3	0
4	1
5	0
6	0

Ilustración 6. Ejemplificación de la problemática, estado final

El resultado obtenido por el proceso de cambios es de un cambio.

Del ejemplo podemos observar, que otra solución posible es:

Estado Final

Alumnos

No. Consecutivo de alumno	Escuela de origen	Escuela solicitada	Calificación	Prioridad	Alumno asignado
4	1	2	40	1	0
6	1	2	40	1	0
3	4	5	36	1	0
2	1	3	32	1	0
7	6	5	31	1	1
1	2	1	22	1	0
5	3	5	22	1	0

Escuelas

No. de escuela	Lugares disponibles
1	0
2	0
3	0
4	0
5	0
6	1

Ilustración 7. Ejemplificación de la problemática, solución alterna 1

El resultado obtenido por el proceso de cambios es de un cambio.

Si el orden descendente de la tabla de alumnos, se sustituye por una ordenación aleatoria se podría presentar lo siguiente:

Estado Final

Alumnos

No. Consecutivo de alumno	Escuela de origen	Escuela solicitada	Calificación	Prioridad	Alumno asignado
2	1	3	32	1	0
5	3	5	22	1	1
4	1	2	40	1	0
6	1	2	40	1	0
3	4	5	36	1	0
7	6	5	31	1	0
1	2	1	22	1	0

Escuelas

No. de escuela	Lugares disponibles
1	0
2	0
3	1
4	0
5	0
6	0

Ilustración 8. Ejemplificación de la problemática, solución alterna 2

El resultado obtenido por el proceso de cambios es de un cambio.

Esta solución proporciona la posibilidad de realizar una o más iteraciones, buscando utilizar los lugares disponibles liberados por los alumnos asignados, con ello se obtendrían mejoras a la solución actual; sin embargo se observa que estas mejoras dependen del orden en que se presenta la tabla de información de alumnos en cada secuencia de asignación.

2.2 PLANTEAMIENTO DEL PROBLEMA

Utilizando la concepción de caja negra [7], para representar el problema que se aborda, se observa que el problema consiste en resolver un proceso de cambios donde se tiene como insumos: Solicitudes de alumnos, lugares disponibles en las escuelas solicitados por los alumnos, alternativas de realización del proceso de cambios y como productos: Resultados del proceso de cambios para cada alumno, confronta de resultados de cada una de las alternativas consideradas y el reporte de lugares disponibles en cada una de las escuelas participantes. En la siguiente ilustración, se realiza una representación del problema a resolver.

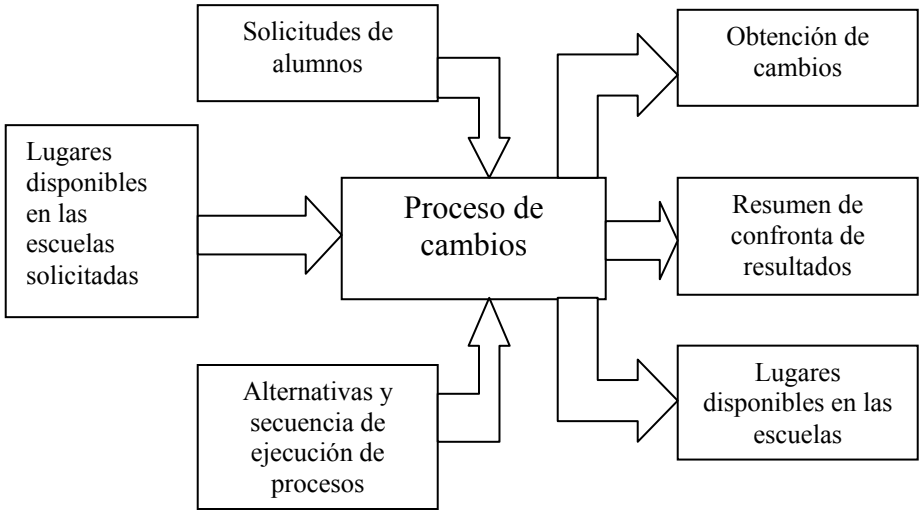


Ilustración 9. Esquema del problema a resolver

2.3 FORMULACIÓN DEL PROBLEMA

Definición de escuelas: Las escuelas participantes son las secundarias oficiales (federales), del Distrito Federal, donde fueron asignados alumnos (proceso de asignación) y las escuelas a donde los alumnos solicitan ser asignados (proceso de cambios).

Definición de participantes: Se considera a todos los aspirantes a ingresar a escuelas secundarias oficiales y que hayan realizado una solicitud de cambio. Estos aspirantes sustentaron un examen donde obtuvieron un puntaje. A la vez, al motivo expresado en la solicitud de cambio se le asigna un peso. Tanto el puntaje como el peso, son utilizados para la asignación de una determinada prioridad a cada aspirante.

Se tiene m escuelas donde se realizan n peticiones de cambio de escuela. Cada petición i se solicita en la escuela deseada r_i, y cada alumno que realiza una solicitud proviene de la escuela e_i. Además, la petición i recibe una prioridad P_i y es realizada por un aspirante con puntaje de aciertos C_i. Cada escuela j cuenta con una disponibilidad d_j de aceptar alumnos, tomando en cuenta que los estudiantes de la escuela j que solicitan cambios y son aceptados dejan nuevos espacios que aumentan la disponibilidad de esta escuela. Además, sea la variable X_i que vale 1 si la petición i se acepta y 0 en caso contrario.

El caso se formula como un problema de programación entera binaria.

Sea: P_i que contiene las prioridades de cada solicitud de cambio; donde $i = 1…n$

Sea: X_i el vector que contiene las variables binarias que representan las solicitudes de cambio; donde $i = 1…n$

Sea: G la función de ganancia a maximizar

Sea d_j el vector que representa la disponibilidad de lugares en cada escuela, $j = 1,\ldots,n$

Entonces el problema se representa a través del siguiente modelo de optimización.

Maximizar

$$G = \sum_{i=1}^{n} P_i X_i$$

Sujeta a:

$$\sum_{i=1}^{k} r_{1,i} - \sum_{j=1}^{l} e_{1,j} \leq d_1$$

.

.

.

$$\sum_{i=1}^{k} r_{m,i} - \sum_{j=1}^{l} e_{m,j} \leq d_m$$

Y

$$X_i = \{0,1\};\ d_j >= \emptyset$$

En esta tesis se utilizan los datos del proceso de cambios del año 2000, donde hubo un total de 12,661 solicitudes y 779 diferentes escuelas participantes.

Dado que el espacio de solución esta dado por todas las combinaciones de cambios tomadas de 1 hasta *n,* el problema aquí presentado se modela en un espacio de estados de tamaño 2^n, para el caso que nos ocupa $2^{12,661}$.

De la formulación se establece que se tienen 12,661 variables binarias con 779 restricciones.

CAPÍTULO 3.
OPTIMIZACIÓN COMBINATORIA

En este capítulo se describe de manera breve la optimización combinatoria y los problemas que resuelve.

3.1 PROBLEMAS COMBINATORIOS

Muchos investigadores han estudiado la forma de encontrar soluciones óptimas a problemas que pueden ser modelados a través de funciones con algunas variables de decisión y tal vez en presencia de algunas restricciones los cuales pueden ser generalmente formulados de la siguiente manera:

Minimizar $f(x)$

sujeta a:

$$g_i(x) \geq b_i; \qquad i = 1,...,m;$$
$$h_j(x) = c_j; \qquad j = 1,...,n.$$

Aquí, x es un vector de variables de decisión, y $f(x)$, $g_i(x)$, $h_j(x)$ son funciones generales, Esta formulación se conoce como un problema de minimización.

Existen muchas clases de estos problemas, obtenidos por la colocación de restricciones sobre el tipo de funciones bajo consideración y sobre los valores que las variables de decisión pueden tomar. Tal vez los más conocidos de esta clase, son aquellos donde las restricciones $f(x)$, $g_i(x)$, $h_j(x)$ son funciones lineales con variables de decisión continuas, los cuales caen en problemas de programación lineal.

Aquí, se consideran una clase especial de problemas, los cuales son de naturaleza combinatoria. Este término es usualmente reservado para casos en donde las variables de decisión son discretas, es decir la solución es un conjunto o secuencia de enteros u otros objetos discretos. El problema de encontrar soluciones óptimas a tales casos es por consiguiente conocido como optimización combinatoria.

Algunos ejemplos son los siguientes:

Ejemplo 1: El problema de asignación, consiste en un conjunto de personas que están disponibles para llevar a cabo un número determinado de tareas; si la persona i es asignada a la tarea j, tiene un costo c_{ij} unidades; la solución consiste en encontrar $\{\pi_1,...,\pi_n\}$ que minimice:

$$\sum_{i=1}^{n} C_{i\pi j}$$

Aquí la solución es presentada por la permutación $\{\pi_1,...,\pi_n\}$, de los números $\{1,...,n\}$.

Ejemplo 2: El problema de la mochila, consiste en un conjunto de n elementos que están disponibles para ser colocados dentro de una mochila con capacidad de C unidades; el elemento i tiene un valor V_i y su uso demanda una capacidad c_i en la mochila. Determinar el subconjunto I de elementos, tal que su inclusión en la mochila maximice:

$$\sum_{i \in I} V_i$$

Tal que:

$$\sum_{i \in I} c_i \leq C$$

Aquí, la solución es representada por el subconjunto $I \subseteq \{1,...,n\}$.

Los problemas de naturaleza combinatoria, tienen fuertes vínculos con la programación lineal, y muchos de los primeros intentos para resolverlos fueron métodos desarrollados a partir de ella, generalmente introduciendo variables enteras que toman valores de 0 ó 1, para producir una formulación de programación entera, por ejemplo, en el caso del problema de la mochila, se define:

$X_i = 1$ si el elemento es empaquetado

$X_i = 0$ si el elemento no es empaquetado

Entonces el problema se traduce en un modelo de programación entera.

Maximizar: $\sum_{i=1}^{n} x_i v_i$

Tal que: $\sum_{i=1}^{n} x_i c_i \leq C$

3.1.1 Óptimos locales y globales

Una de las características de algunos problemas combinatorios es que pueden tener muchos óptimos globales, mientras que pueden tener más óptimos locales. Entiéndase por óptimo global la combinación que maximiza el valor de la función de ganancia. El proceso de búsqueda de la solución óptima puede consistir de una o más iteraciones, el mejor valor encontrado en cada iteración es conocido como un óptimo local. Esto se puede entender mejor, si se introduce el concepto de vecindad.

Una vecindad $N(x,\sigma)$ de una solución $\boldsymbol{x}$ es un conjunto de soluciones que pueden ser alcanzadas a partir de $\boldsymbol{x}$ aplicando una operación σ. Tal operación σ puede ser agregando o quitando un elemento de la solución. El intercambio de objetos es otro ejemplo de tal operación, éstos son particularmente comunes en problemas de secuenciación. En la técnica BT, estas operaciones son llamadas movimientos. Si una solución $\boldsymbol{y}$ es mejor que cualquier otra solución en esta vecindad $N(x,\sigma)$, entonces $\boldsymbol{y}$ es un óptimo local con respecto a esa vecindad.

En algunos casos es posible encontrar un movimiento σ, tal que un óptimo local sea también un óptimo global, por ejemplo, en algunos casos de problemas de secuenciación de máquinas, el intercambio de un par de elementos tiene esta propiedad.

La mayoría de los tratamientos existentes para los problemas de optimización combinatoria, se concentran en métodos exactos, más que en heurísticos. Estos métodos están ligados a la teoría de la programación lineal o usan tecnicas de enumeración implícita.

3.1.2 Técnicas Heurísticas

El término heurística, deriva del griego heuriskein que significa encontrar o descubrir. Una técnica heurística, es aquella que encuentra buenas soluciones, a un costo computacional razonablemente aceptable, sin que éstas sean necesariamente las óptimas.

Una gran cantidad de artículos tratan de cómo las técnicas heurísticas se han utilizado para resolver problemas de optimización combinatoria. Esto puede ser atribuido a dos razones:

1. Por un lado, el concepto de complejidad computacional ha proporcionado las bases para explorar las técnicas heurísticas.

2. Por otro lado, han surgido nuevas técnicas más eficientes para resolver problemas de optimización combinatoria en tiempos de cómputo razonables.

3.2 PROBLEMAS NP COMPLETOS

Problema del agente viajero: Un vendedor partiendo de una ciudad tiene que visitar *n-1* ciudades diferentes y regresar al punto de partida. Si el costo de dirigirse a la ciudad *j* desde la ciudad *i*, es c_{ij} $(c_{ij} \neq c_{ji})$, se debe determinar la secuencia de visita de ciudades que le permita visitar cada una de las *n* ciudades una y solo una vez, tal que el costo asociado sea mínimo. Hay *(N-1)!* Posibles soluciones o $\frac{(N-1)!}{2}$ si la distancia entre cada par de ciudades es la misma sin considerar la dirección del viaje.

Supóngase que tenemos una computadora que puede listar todas las posibles soluciones de un ejemplar; entonces utilizando la fórmula anterior, tendríamos como resultado lo presentado en la ilustración 10, que nos muestra el crecimiento del tiempo de cómputo con respecto al tamaño de la entrada del problema, en la tabla podemos observar que la enumeración completa es ineficiente para obtener una solución óptima, ya que por ejemplo, considerando una determinada computadora que utiliza una hora para resolver el problema de 20 ciudades, un problema de 25 ciudades tomaría aproximadamente 6 siglos en ser resuelto.

Número de ciudades	Tiempo de cómputo
20	1 hora
21	20 horas
22	17.5 días
23	1.05 años
24	24.26 años
25	5.82 siglos

Ilustración 10. Tiempo de cómputo demandado por el problema del agente viajero

Varios algoritmos exactos han sido inventados para encontrar soluciones óptimas de problemas de manera más eficiente que la enumeración completa. El algoritmo más conocido es el Simplex para problemas de programación lineal. Estos algoritmos son capaces de resolver pequeños ejemplares, pero no para encontrar soluciones óptimas, en una cantidad de tiempo razonable computacional, cuando los ejemplares son grandes. Como el poder computacional se ha incrementado en los últimos años, ha sido posible resolver grandes problemas, y los investigadores se han interesado en cómo el tiempo de solución varía con el tamaño de la entrada del problema.

Para problemas tales como el del agente viajero, el esfuerzo computacional sigue siendo exponencial. A finales de los años 60´s los investigadores se hacían la siguiente pregunta. ¿Habrá un algoritmo de optimización en tiempo polinomial para un problema como el del agente viajero?. Nadie ha sido capaz de responder a esta pregunta, sin embargo en 1972, Karp [8], mostró que si la respuesta fuera afirmativa para el problema del agente viajero entonces habría también un algoritmo en tiempo polinomial para otros problemas equivalentes. Como ningún algoritmo ha sido encontrado para estos problemas, esto indica que la respuesta a la pregunta original es probablemente no. Sin embargo, la respuesta real es desconocida.

Hay problemas que pueden ser resueltos con algoritmos polinomiales conocidos y se dice que pertenecen a la clase P. Pero ¿qué hay del problema del agente viajero y otros problemas equivalentes? ¿Son de complejidad exponencial? Muchos de estos problemas están en la clase NP, que es una abreviación de non-deterministic polinomial. De hecho el problema del agente viajero es de los problemas más difíciles en NP. Sí un algoritmo polinomial fuera encontrado para este problema, significaría que existe un algoritmo polinomial para los demás problemas NP; pero como ningún algoritmo polinomial exacto ha sido encontrado para cualquier problema en NP, hay fuerte evidencia de que **P <> NP**, lo cual es un argumento para buscar formas alternativas de resolver problemas computacionalmente difíciles. Existe la posibilidad

de que alguien llegue a probar que **P = NP**; pero mientras no se encuentre tal prueba, el uso de las técnicas heurísticas se justifica.

Existen otros argumentos a favor del uso de las técnicas heurísticas. Lo que realmente se está optimizando es un modelo de un problema del mundo real, por eso no hay garantía de que la mejor solución del modelo sea también la mejor solución para el problema del mundo real. Las técnicas heurísticas son usualmente más flexibles y son capaces de hacer lo mismo con funciones objetivo y/o restricciones más complicadas que los algoritmos exactos. Este es el caso de técnicas tales como la Búsqueda Tabú; donde las funciones objetivo no necesitan cumplir hipótesis de linealidad. Esto nos permite modelar problemas del mundo real aún más precisamente que con el uso de algoritmos exactos.

Muchos problemas de optimización combinatoria son específicos, de manera que un algoritmo de una técnica heurística que funciona para un caso, puede no ser útil para resolver un problema diferente. Sin embargo, hay un gran interés en técnicas que se han desarrollado en la última década y que pueden ser aplicables con mayor generalidad.

Algunas de estas técnicas han sido desarrolladas bajo la búsqueda local de vecinos (local neighborhood search). El proceso inicia con una solución para un ejemplar, buscándose una mejor solución dentro de una vecindad definida. Habiendo encontrado una mejor solución, el proceso reinicia la búsqueda local con esta nueva solución. El proceso continúa iterativamente hasta que no pueda mejorar la solución actual encontrada. Esta solución final, probablemente sea un óptimo local.

Una variación a lo anterior que ha ganado reciente atención es el permitir movimientos ascendentes, con lo cual la solución con la que la búsqueda local reinicia puede ser peor que la anterior, considerando que debe haber algunas restricciones para aceptar

tales movimientos, en otro caso el procedimiento se sumaría a la búsqueda del espacio completo de soluciones. Por lo tanto se da la oportunidad de que el proceso pueda escapar o salir de óptimos locales y busque una mejor solución.

CAPÍTULO 4.
MARCO TEÓRICO DE BÚSQUEDA TABÚ

En este capítulo se expone el origen del meta-heurístico Búsqueda Tabú y se explica en que consiste. Conviene señalar que se pueden generar diversas variantes del método a partir de la inclusión de conceptos de otros métodos heurísticos.

4.1 INTRODUCCIÓN

El diccionario Enciclopédico Master, define tabú como: “lo relacionado con la prohibición de carácter mágico religioso que puede afectar a personas, lugares, cosas, o circunstancias. Es propia de los pueblos primitivos y tiene fundamento animista. Lo que es tabú no puede tocarse y a veces, ni siquiera ser mirado o nombrado. Puede tener doble aspecto: el de cosa sagrado o el de cosa maléfica”.

En computación, un algoritmo que produce con rapidez soluciones buenas, pero no necesariamente óptimas, se denomina heurístico [9]. En aplicaciones de optimización la búsqueda de soluciones óptimas, puede demandar búsquedas exhaustivas considerando todas las posibilidades, implicando en ocasiones costos computacionales muy elevados. Sin embargo partiendo del enfoque de métodos heurísticos y meta-heurísticos, se puede aspirar a encontrar buenas soluciones aunque éstas no necesariamente sean las óptimas.

El término meta-heurístico fue acuñado en la misma publicación [8] que introdujo el término Búsqueda Tabú, y ha sido ampliamente aplicado en la literatura; un meta-heurístico se refiere a una estrategia maestra que guía y modifica otros heurísticos para producir soluciones más allá de las que normalmente se generan en la búsqueda de óptimos locales.

El contraste entre la orientación de un meta-heurístico y el de óptimo local es significante. Por muchos años las primeras concepciones de un procedimiento heurístico, fue considerar alguna regla inteligente iterativa que terminaba cuando no obtenía de manera inmediata soluciones que mejoraran la inmediata anterior. Tales heurísticos iterativos son con frecuencia referidos como métodos ascendente, descendente y búsqueda local.

En los últimos seis años los meta-heurísticos han evolucionado habiendo experimentando considerables mejoras. En su forma moderna los meta-heurísticos se basan sobre una variedad de interpretaciones que constituyen búsquedas "inteligentes". Esto puede ser ilustrado en términos de las características de tres opciones básicas de diseño:

1.- Memoria adaptativa,
2.- La clase de exploración de vecinos utilizada, y
3.-El número de soluciones llevadas de una iteración a otra.

La memoria adaptativa en BT permite la implementación de procedimientos que son capaces de buscar en el espacio de solución de manera económica y efectiva. Si estas opciones son clasificadas como x/y/z donde las opciones para 'x' son A (si el meta-heurístico emplea memoria adaptativa) y M si el método utiliza memoria corta (se realiza la búsqueda en un subconjunto de la vecindad de soluciones a una solución dada). Para el caso de '*y*' son *N* (para métodos que emplean búsqueda sistemática de vecinos para seleccionar el siguiente movimiento o mejorar una solución dada) y *S* (para métodos que dependen de muestras aleatorias), finalmente para '*z*' pueden ser 1 (si el método se mueve de una solución a otra después de cada iteración) o *P* (para un acceso basado en poblaciones con una población de tamaño *P*). Este esquema tridimensional proporciona una base preliminar de clasificación lo cual revela que el acuerdo para etiquetar varios meta-heurísticos está lejos de ser uniforme.

Con base en la información anterior, en la siguiente ilustración se presenta la clasificación de los métodos heurísticos más conocidos.

Meta-Heurístico	Clasificación 1	Clasificación 2
Algoritmos Genéticos	M/S/P	M/N/P
Recocido Simulado	M/S/1	M/N/1
Búsqueda Tabú	A/N/1	A/N/P

Ilustración 11. Clasificación de los métodos heurísticos

Las dos clasificaciones son dadas para cada procedimiento, la primera está más apegada a la concepción popular y la segunda es dada por un número significativo de investigadores.

Actualmente algunos seguidores del método del Recocido Simulado y Algoritmos Genéticos están modificando el concepto original de estos métodos e introduciendo elementos de memoria adaptativa tal y como la usa Búsqueda Tabú.

La forma básica de BT, está fundada sobre las ideas propuestas por Fred Glover [8]. El método se basa en procedimientos para cruzar límites de factibilidad u óptimos locales tratados usualmente como barreras. Sistemáticamente impone y libera restricciones para permitir la exploración de otras regiones.

Búsqueda Tabú es un meta-heurístico que guía a un procedimiento de búsqueda heurística local a explorar el espacio de solución más allá del óptimo local. El procedimiento de búsqueda local, utiliza una operación llamada *movimiento* para definir la vecindad de una solución.

La filosofía de BT se basa en derivar y explotar una colección de principios inteligentes en la solución de problemas [10]. Un elemento fundamental es el uso de memoria flexible, la cual desde el punto de vista de BT envuelve el proceso dual de

crear y explotar estructuras para tomar ventaja de la historia combinando las actividades de adquisición y obtención de provecho de la información.

4.1.1 Descripción de Búsqueda Tabú

Para describir el método BT representamos el problema de optimización combinatoria de la siguiente forma:

Minimizar $c(x)$

Sujeta a $x \in X$.

La función objetivo $c(x)$ puede ser lineal o no lineal, y la condición $x \in X$, supone que las restricciones especificadas de los componentes de x sean valores discretos.

4.1.2 Búsqueda local de vecinos

Búsqueda Tabú inicia de manera similar a la búsqueda local ordinaria o búsqueda local de vecinos, procediendo de manera iterativa de un punto (solución) a otro, hasta que un determinado criterio de terminación es satisfecho. Cada $x \in X$ tiene asociado una vecindad denominada $V(x) \subset X$, y cada solución $x' \in V(X)$ es accesada desde x por una operación llamada *movimiento* se dice que x se mueve o transita a x' cuando el movimiento es realizado. Normalmente se dice que x y x' son simétricos si y solo si x' es vecino de x.

4.1.3 Método de búsqueda local de vecinos

1. Inicialización:

   a) Seleccione una solución inicial $x^{act} \in X$

   b) Registre la solución en cuestión como la mejor solución $x^{mejor} = x^{act}$ y $mejor\_costo = c(x^{mejor})$.

2. Selección y terminación:

   Elija una solución x^{sig} de $V(x^{act})$, si el criterio empleado no puede ser satisfecho por cualquier miembro de $V(x^{act})$ (ninguna solución califica como x^{sig}), o si otro criterio de terminación es satisfecho (tal es el caso del límite de iteraciones), entonces el método termina.

3. Actualización:

   Actualice $x^{act} = x^{sig}$, y si $c(x^{act}) < mejor\_costo$, ejecute 1(b) y entonces prosiga con el paso 2.

En los pasos de búsquedas de vecinos descritos, se asume que existen criterios para seleccionar movimientos y criterios para la terminación de la búsqueda y que son dados por un conjunto de prescripciones externas.

La búsqueda local de vecinos puede ser modificada para generar los métodos descendentes, en estos sólo se permiten movimientos a soluciones vecinas que mejoren el valor actual de $c(x^{act})$, y terminan cuando la solución actual no puede ser mejorada.

Un método descendente genérico puede representarse mediante el siguiente seudo-código.

Método descendente:

1. Inicialización:

   Empezar con la inicialización de la búsqueda local de vecinos (Elija $x \in X$ para iniciar el proceso).

1. Selección y terminación:

   Escoger $x^{sig} \in V(x^{act})$ para satisfacer $c(x^{sig}) < c(x^{act})$, Termina si x^{sig} no puede ser encontrada.

2. Actualización:

   Realizar la actualización de la búsqueda local de vecinos.

Ilustración 12. Método descendente

La *x* final obtenida por un método descendente es llamada un óptimo local, ya que es tan buena o mejor que todas las soluciones vecinas. Es evidente que este resultado en la mayoría de los casos no corresponde al óptimo global y por ende no minimiza *f(x)* sobre el conjunto solución $x \in X$.

4.2 CARACTERÍSTICAS DE LA BÚSQUEDA TABÚ

BT utiliza una filosofía diferente a los métodos tradicionales de búsqueda de óptimos locales. La aleatoriedad es empleada en la generación de soluciones en determinadas situaciones, ya que se parte del principio de utilizar búsquedas inteligentes basadas en procedimientos sistemáticos, implicando que en su mayoría las implementaciones de BT sean determinísticas.

La relevancia de elegir buenas soluciones de un conjunto de vecindades es magnificada cuando el mecanismo de BT ha ido más allá del punto de terminación del óptimo local obtenido con el método descendente; así, un importante nivel de consideración para Búsqueda Tabú es determinar un apropiado conjunto de estrategias para reducir la revisión de elementos de $V(X)$, para lograr una efectiva búsqueda entre la calidad de x' y el esfuerzo utilizado para alcanzarla.

La noción de explotar ciertas formas de memoria flexible para controlar los procesos de búsqueda, es el tema central de BT; el efecto de tales formas de memoria puede ser imaginado estipulando que el método mantiene una historia selectiva H, de los estados encontrados durante la búsqueda y reemplaza $V(x^{act})$ por una vecindad modificada la cual puede ser denotada $V(H, x^{act})$, la historia determina por lo tanto cuales soluciones pueden ser alcanzadas por un movimiento a partir de una solución dada, seleccionando x^{sig} de $V(H, x^{act})$.

Las estrategias del método de BT, basadas sobre consideraciones de términos de memoria corta $V(H, x^{act})$, es típicamente un subconjunto de $V(x^{act})$, y la clasificación tabú, sirve para identificar elementos de $V(x^{act})$ excluidos de $V(H, x^{act})$.

En las estrategias de memoria intermedia y larga, la vecindad $V(H, x^{act})$ puede incluir soluciones no contenidas en $V(x^{act})$, generalmente consiste en seleccionar una elite de soluciones seleccionadas (óptimos locales de alta calidad), encontrados en varios puntos del proceso de solución.

BT también utiliza la historia para crear una evaluación modificada de la solución accesible actual. Formalmente esto puede ser expresado diciendo que la BT reemplaza la función objetivo $c(x)$ por una función $c(H, x)$, lo cual tiene como propósito evaluar la calidad relativa de las soluciones encontradas. La relevancia de esta función modificada ocurre porque Búsqueda Tabú usa selectos criterios agresivos que buscan una mejor x^{sig}. Es decir el mejor valor de $c(H, x^{sig})$, sobre el conjunto de candidatos obtenidos de $V(H, x^{act})$. La referencia a $c(x)$, es retenida para determinar si un movimiento nos conduce a una mejor solución.

Para problemas grandes, donde $V(H, x^{act})$ puede tener muchos elementos o para problemas donde puede ser costoso examinar dichos elementos, la orientación agresiva de BT hace altamente importante aislar un subconjunto candidato de la vecindad para ser examinados en vez de toda la vecindad. Esto puede ser realizado en etapas, permitiendo que los subconjuntos de candidatos sean expandidos si las alternativas de niveles de aspiración no son encontradas.

En BT ubicamos tres etapas fundamentales:

1. Etapa de inicialización: Se genera una solución inicial. Esta puede ser factible o no y se evalúa la solución objetivo.
2. Etapa de movimientos: A través de intercambio de parejas de datos se generan soluciones adyacentes de una solución determinada. De éstas en caso de existir se selecciona la mejor .

3. Se ejecuta el movimiento y se realiza la actualización: En la evaluación de un vecindario si existe una mejor solución a la de referencia, se realiza el desplazamiento a esa punto adyacente y se actualiza el valor de la función objetivo.

4.3 MÉTODO BÁSICO DE LA BÚSQUEDA TABÚ

El método básico de Búsqueda Tabú se describe en la siguiente ilustración.

Búsqueda Tabú:

Paso 1: Inicialización:

Generar una solución inicial que será utilizada para la búsqueda de vecinos e iniciar con cero, cada uno de los registros de la historia *H*.

Paso 2: Selección y terminación:

- Determinar el subconjunto $candidato\_V(x^{act})$ como un subconjunto de $V(H, x^{act})$.
- Seleccionar $x^{sig} \in candidato\_V(x^{act})$ para minimizar $c(H,x)$ sobre este subconjunto, (x^{sig} es llamado el mejor elemento, evaluado de $candidato\_V(x^{act})$).
- Terminar por el número de iteraciones si no se encuentra una solución óptima o si no se satisface la regla de finalización.

Paso 3: Actualización:

Realizar la actualización por el método de búsqueda de vecinos y adicionalmente actualizar el registro de la historia *H*.

Ilustración 13. Método básico de la Búsqueda Tabú

4.3.1 Memoria de la Búsqueda Tabú

La estructura de la memoria de la BT opera por referencia a 4 dimensiones principales.

1. Los movimientos más recientes (*recency), m*emoria que guarda los movimientos efectuados recientemente y que no serán elegidos nuevamente hasta salir de esta memoria (lista tabú).

2. Frecuencia (*frequency*), número total de ocurrencia de todos los eventos.

3. Calidad (*quality), v*alor del movimiento efectuado.

4. Influencia (*influence*), mide el grado de cambio inducido en una solución.

Un atributo de un movimiento de x^{act} a x^{sig} o más precisamente de un movimiento de prueba x^{act} a una solución tentativa x^{prueba}, puede abarcar cualquier aspecto que cambie como resultado del movimiento. Por ejemplo, un movimiento que cambia el valor de dos variables simultáneamente.

El registro de los atributos de un movimiento es en ocasiones utilizado en la BT para imponer restricciones, llamadas restricciones tabú, que evitan elegir movimientos que inviertan los cambios presentados por estos atributos. Por ejemplo, un movimiento es Tabú si cambia de 1 a 0, donde x_j cambió de 0 a 1.

Las restricciones tabú también son utilizadas para evitar movimientos que conduzcan a repeticiones o ciclos en un camino de búsqueda. Estas restricciones tienen el papel de evitar el regreso a un movimiento cuyos atributos produzcan una solución previamente encontrada. Por lo tanto las restricciones tabú varían de acuerdo a como son definidas.

Una restricción tabú es activada únicamente en el caso de que sus atributos ocurran dentro de cierto número de iteraciones anteriores a la iteración actual, creando la lista tabú de los movimientos más recientes; también, las restricciones tabú pueden ser activadas cuando los atributos se presentan con cierta frecuencia a lo largo de las iteraciones, creando la matriz de frecuencia. Una restricción tabú es ejecutada únicamente cuando los atributos de la definición rebasen ciertos umbrales de frecuencia o sean iguales a los movimientos más recientes. Para entender esto, se define un atributo como tabú-activo cuando su atributo inverso asociado ha ocurrido en cierto intervalo de los movimientos mas recientes. Un atributo que no es activo es llamado tabú inactivo.

La condición de ser tabú-activo o tabú-inactivo es llamada el estado tabú de un atributo. En algunas ocasiones un atributo es llamado tabú o no-tabú para indicar que es tabú-activo o tabú-inactivo.

La determinación del tiempo "*t*" que un movimiento es tabú puede realizarse de dos maneras:

a) Considerando reglas estáticas, donde el valor "*t*" permanece fijo a través de la búsqueda. Elegir "*t*" constante en muchos casos prácticos, se le asigna el valor de 7, o $\sqrt{n}$, donde *n* es la dimensión del problema.

b) Considerando reglas dinámicas, donde se permite que el valor de "*t*" varíe. Esto es, la "*t*" puede estar entre dos valores ó limites t_{min} y t_{max}, donde t_{min} puede ser de 5 y t_{max} puede ser igual a 11, ó $t_{min} = .9\sqrt{n}$ y $t_{max} = 1.1\sqrt{n}$.

La experiencia práctica indica que la regla dinámica es más robusta que la regla estática.

4.3.2 Criterios de aspiración

Los criterios de aspiración se introducen en la técnica de BT para poder determinar cuándo una restricción tabú puede ser rechazada, eliminando la clasificación de tabú aplicada a cierto movimiento. El uso apropiado de este criterio de aspiración puede resultar muy importante para permitir así un buen nivel de desempeño de la BT.

Las primeras aplicaciones de la BT emplearon únicamente un tipo simple del criterio de aspiración, el cual consistía en rechazar la clasificación tabú de un movimiento de prueba cuando al evaluar la función objetivo se obtiene una mejor solución que la mejor solución encontrada hasta el momento. Este criterio de aspiración sigue siendo utilizado en muchas aplicaciones, sin embargo, otros criterios pueden mejorar también la efectividad de la Búsqueda Tabú.

Uno de los criterios de aspiración surge introduciendo el concepto de influencia, el cual mide el grado de cambio inducido en la estructura de la solución, la influencia es en ocasiones asociada con la idea de la distancia del movimiento, es decir, un movimiento de mayor distancia es concebido como el de mayor influencia.

Los movimientos de gran influencia son importantes especialmente durante los intervalos para romper con la optimalidad local, debido a que una serie de movimientos que hacen solo pequeños cambios estructurales es improbable que descubran un mejoramiento importante.

Las aspiraciones son de dos tipos: movimientos de aspiración y atributos de aspiración. Un movimiento de aspiración, cuando se satisface, rechaza el estado tabú del movimiento. Un atributo de aspiración, cuando se satisface, rechaza el estado tabú del atributo. En el último caso, el movimiento pudo o no cambiar su clasificación tabú, dependiendo si la restricción tabú puede ser activada por más de un atributo.

4.4 EJEMPLIFICACIÓN DEL FUNCIONAMIENTO DE BÚSQUEDA TABÚ

Supóngase que se cuenta con una mochila que soporta un peso total de 13 Kg., se desea transportar en dicha mochila, 5 objetos con los siguientes pesos en Kilogramos 8,7,12,4 y 2. ¿Qué elementos deben incorporarse para hacer uso óptimo de la mochila?.

Consideraciones:

1.- El movimiento a utilizar será el introducir o sacar un elemento a la vez de la mochila.
2.-El tiempo que un movimiento será tabú, se obtendrá de manera dinámica partiendo de una lista de números aleatorios y estará en el rango de 1 a 5 iteraciones.
3.-En caso de que un movimiento sea tabú, y éste conduzca a la solución óptima, dicha restricción tabú será rechazada.

Estado inicial:

Elemento	1	2	3	4	5
Peso del elemento	8	7	12	4	2
Inclusión en la mochila	0	0	0	0	0
Peso total incluido en la mochila	0				
Capacidad de la mochila	13				

Registro	Tiempo tabú
1	0
2	0
3	0
4	0
5	0

Lista de números aleatorios
3
5
1
2
4

Ilustración 14. Funcionamiento de Búsqueda Tabú, estado inicial.

Paso inicial (se genera una solución factible):

Elemento	1	2	3	4	5
Peso del elemento	8	7	12	4	2
Inclusión en la mochila	1	0	0	1	0
Peso total incluido en la mochila	12				

Registro	Tiempo tabú
1	0
2	0
3	0
4	0
5	0

Lista de números aleatorios
3
5
1
2
4

Ilustración 15. Funcionamiento de Búsqueda Tabú, solución inicial factible

Primera iteración:

Elemento	1	2	3	4	5
Peso del elemento	8	7	12	4	2
Inclusión en la mochila	***0***	0	0	1	0
Peso total incluido en la mochila	4				
	4				

Registro	Tiempo tabú	Lista de números aleatorios
1	3	3
2	0	5
3	0	1
4	0	2
5	0	4

Elemento	1	2	3	4	5
Peso del elemento	8	7	12	4	2
Inclusión en la mochila	0	***1***	0	1	0
Peso total incluido en la mochila	11				

Registro	Tiempo tabú	Lista de números aleatorios
1	3	3
2	5	5
3	0	1
4	0	2
5	0	4

Elemento	1	2	3	4	5
Peso del elemento	8	7	12	4	2
Inclusión en la mochila	0	1	***0***	1	0
Peso total incluido en la mochila	11				

Registro	Tiempo tabú	Lista de números aleatorios
1	3	3
2	5	5
3	0	1
4	0	2
5	0	4

Elemento	1	2	3	4	5
Peso del elemento	8	7	12	4	2
Inclusión en la mochila	0	1	0	***0***	0
Peso total incluido en la mochila	7				

Registro	Tiempo tabú	Lista de números aleatorios
1	3	3
2	5	5
3	0	1
4	1	2
5	0	4

Elemento	1	2	3	4	5
Peso del elemento	8	7	12	4	2
Inclusión en la mochila	0	1	0	0	***1***
Peso total incluido en la mochila	9				

Registro	Tiempo tabú	Lista de números aleatorios
1	3	3
2	5	5
3	0	1
4	1	2
5	2	4

Ilustración 16. Funcionamiento de Búsqueda Tabú, primera iteración

Segunda iteración

¡Er

Elemento	1	2	3	4	5
Peso del elemento	8	7	12	4	2
Inclusión en la mochila	***0***	1	0	0	1
Peso total incluido en la mochila	9				

Registro	Tiempo tabú
1	3
2	5
3	0
4	1
5	2

Lista de números aleatorios
3
5
1
2
4

En la siguiente evaluación, obsérvese que aún cuando el movimiento del elemento dos es factible de hacerse, éste no se realiza ya que al estar en la lista tabú, es prohibido.

¡Error!

Elemento	1	2	3	4	5
Peso del elemento	8	7	12	4	2
Inclusión en la mochila	0	***1***	0	0	1
Peso total incluido en la mochila	9				

Registro	Tiempo tabú
1	3
2	5
3	0
4	1
5	2

Lista de números aleatorios
3
5
1
2
4

Elemento	1	2	3	4	5
Peso del elemento	8	7	12	4	2
Inclusión en la mochila	0	1	***0***	0	1
Peso total incluido en la mochila	9				

Registro	Tiempo tabú
1	3
2	5
3	0
4	1
5	2

Lista de números aleatorios
3
5
1
2
4

En esta iteración, aun cuando el movimiento del elemento cuatro, es prohibido, Búsqueda Tabú puede romper esta prohibición ya que el hacerlo implica encontrar el óptimo (criterio de aspiración), sin embargo para realizar la ejemplificación completa esto no se realizará.

Elemento	1	2	3	4	5
Peso del elemento	8	7	12	4	2
Inclusión en la mochila	0	1	0	***0***	1
Peso total incluido en la mochila	9				

Registro	Tiempo tabú
1	3
2	5
3	0
4	1
5	2

Lista de números aleatorios
3
5
1
2
4

Obsérvese que aún cuando el movimiento del elemento cinco es factible de hacerse, éste no se realiza ya que al estar en la lista tabú, es prohibido.

Elemento	1	2	3	4	5
Peso del elemento	8	7	12	4	2
Inclusión en la mochila	0	1	0	0	***1***
Peso total incluido en la mochila	9				

Registro	Tiempo tabú
1	3
2	5
3	0
4	1
5	2

Lista de números aleatorios
3
5
1
2
4

Ilustración 17. Funcionamiento de Búsqueda Tabú, segunda iteración

Tercera iteración:

Elemento	1	2	3	4	5
Peso del elemento	8	7	12	4	2
Inclusión en la mochila	***0***	1	0	0	1
Peso total incluido en la mochila	9				

Registro	Tiempo tabú
1	2
2	4
3	0
4	0
5	1

Lista de números aleatorios
3
5
1
2
4

Elemento	1	2	3	4	5
Peso del elemento	8	7	12	4	2
Inclusión en la mochila	0	***1***	0	0	1
Peso total incluido en la mochila	9				

Registro	Tiempo tabú
1	2
2	4
3	0
4	0
5	1

Lista de números aleatorios
3
5
1
2
4

Elemento	1	2	3	4	5
Peso del elemento	8	7	12	4	2
Inclusión en la mochila	0	1	***0***	0	1
Peso total incluido en la mochila	9				

Registro	Tiempo tabú
1	2
2	4
3	0
4	0
5	1

Lista de números aleatorios
3
5
1
2
4

Elemento	1	2	3	4	5
Peso del elemento	8	7	12	4	2
Inclusión en la mochila	0	1	0	***1***	1
Peso total incluido en la mochila	13				

Registro	Tiempo tabú
1	3
2	5
3	0
4	4
5	1

Lista de números aleatorios
3
5
2
1
4

Ilustración 18. Funcionamiento de Búsqueda Tabú, tercera iteración

Del ejemplo puede observarse que en la iteración dos y tres se encuentra la combinación de elementos óptima, a incluir en la mochila.

CAPÍTULO 5.
IMPLEMENTACIÓN

En este capítulo se describe el diseño y la programación del modelo de solución.

5.1 CONSIDERACIONES

Se diseña y construye una aplicación informática que contiene:

- ✓ La programación del método tradicional utilizado en la solución del problema de cambios y el cual es empleado a la fecha.

- ✓ La programación de una variante del método tradicional descrito en el inciso anterior, y que consiste en ejecutar recurrentemente el método tradicional mientras se realice al menos un cambio por iteración.

- ✓ La programación del meta-heurístico Búsqueda Tabú.

El desarrollo Informático se realizó utilizando la herramienta de programación Borland Inprise Delphi versión 5.0.

5.1.1 Descripción

El diseño de la aplicación utiliza el paradigma de programación orientada a objetos, donde se tienen cuatro clases ver ilustración 19.

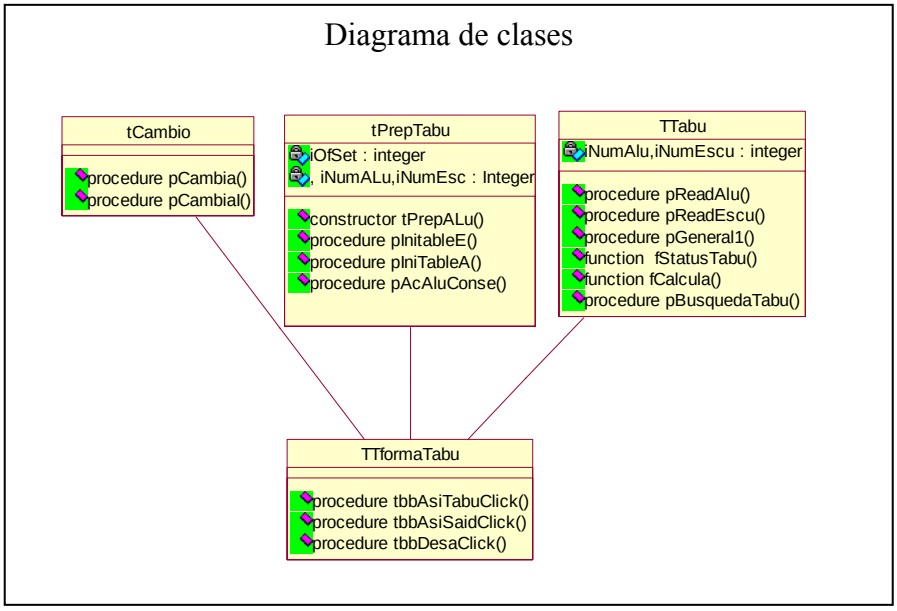


Ilustración 19. Diagrama de clases que integran la solución

Clase TCambio: Realiza el proceso de asignación tradicional de cambios de escuela y contiene los siguientes métodos:

1. Procedure pCambia(tALu, tEscuela): Realiza el proceso tradicional de cambios. Se recorre secuencialmente la tabla de alumnos, para cada alumno en cuestión, en caso de existir lugares disponibles en la escuela destino, actualiza a uno la asignación del alumno, se decrementa en uno la disponibilidad de lugares en la escuela destino y se incrementa en uno los lugares disponibles en la escuela de origen.

2. Procedure pCambiaI(): Realiza el proceso de desasignación. Se recorre secuencialmente la tabla de alumnos, en los casos donde los alumnos fueron asignados, se inicializa a cero su asignación, se decrementa en uno la disponibilidad de lugares en la escuela de origen y se incrementa en uno los lugares disponibles en la escuelas destino.

Clase TPrepTabu: Realiza la transformación de datos de la aplicación informática actual al formato utilizado por la aplicación informática desarrollada en esta tesis y contiene los siguientes métodos:

1. Procedure tPreAlu(): Método constructor de la clase, inicializa el valor del atributo iOfset que es utilizado para dar una sola clave a cada escuela.

2. Procedure pIniTableE(): Prepara la tabla de información de escuelas que será utilizada por la clase TTabu.

3. Procedure pIniTableA(): Prepara la tabla de información de alumnos que será utilizada por la clase TTabu.

4. Procedure pActAluConse(): Actualiza las claves de escuela origen y destino de cada registro de la tabla de información de alumnos.

Clase TTabu: Esta es la clase principal del sistema, sus métodos contienen la programación del método Búsqueda Tabú.

1. Procedure pReadAlu(): Realiza la lectura de los registros de la tabla de alumnos.

2. Procedure pReadEscu(): Realiza la lectura de los registros de la tabla de escuelas.

3. Procedure pGeneral1(): Genera una solución inicial factible, para ello utiliza un recorrido secuencial aleatorio de los registros de información de alumnos participantes.

4. Function fStatusTabu(): Revisa que un movimiento no sea tabú (prohibido).

5. Function fCalcula(): Calcula el valor de una vecindad (solución factible).

6. Procedure pBusquedaTabu(): Realiza las *n* iteraciones de la Búsqueda Tabú teniendo las siguientes consideraciones.

   - ✓ El número máximo de iteraciones es de 1,000,000.
   - ✓ El valor de aspiración de la función de ganancia es de 50,000.
   - ✓ El criterio de paro es por número de iteraciones o alcanzar el valor de aspiración de la función de ganancia.
   - ✓ El tiempo tabú de cada movimiento es dinámico y se determina de manera aleatoria encontrándose en el rango de 15 a 20.
   - ✓ En cada iteración la búsqueda de movimientos factibles se realiza evaluando de manera aleatoria todos los registros de alumnos.

CAPÍTULO 6.
RESULTADOS OBTENIDOS

En este capítulo se presentan los resultados obtenidos, las conclusiones y contribuciones principales de esta tesis.

6.1 RESULTADOS EXACTOS DE BÚSQUEDA TABÚ CON MUESTRA DE DATOS

Comparación de los resultados obtenidos en la solución de una muestra de datos, a través del método exacto y la programación de Búsqueda Tabú.

Para ilustrar los resultados que la aplicación obtiene, se presentan dos ejecuciones del programa con una muestra pequeña de datos, para el primer caso todos los aspirantes a cambio tienen prioridad 1, para el segundo caso existe un alumno que tiene prioridad 2.

1.- Primer caso, alumnos sin prioridad. La solución óptima es de cuatro cambios, siendo dos posibles combinaciones – Cambio a los alumnos 1,2,5,4 ó 1,2,5,6 –.

a) Muestra de datos de alumnos para la ejecución del heurístico Búsqueda Tabú programado, sin utilizar prioridad de asignación.

No. Consecutivo de alumno	Escuela de origen	Escuela solicitada	Calificación	Prioridad
1	2	1	22	1
2	1	3	32	1
3	4	5	36	1
4	1	2	40	1
5	3	5	22	1
6	1	2	40	1
7	6	5	31	1

Ilustración 20. muestra de datos de alumnos para la ejecución de Búsqueda Tabú, sin prioridad de asignación

c) Muestra de datos de escuelas para la ejecución del heurístico Búsqueda Tabú programado, sin utilizar prioridad de asignación.

No. de escuela	Lugares disponibles
1	0
2	0
3	0
4	0
5	1
6	0

Ilustración 21. muestra de datos de escuelas para la ejecución de Búsqueda Tabú, sin prioridad de asignación

c) Proceso de asignación utilizando Búsqueda Tabú

El estado de cambios de alumnos es representado por la secuencia de ceros y unos de izquierda a derecha a saber, el primer 0 representa al alumno 1, el segundo 0 representa al alumno 2 y así sucesivamente.

La lista Tabú, esta representada por valores en el rango de tiempo 0 a 8, la primera posición de izquierda a derecha representa al alumno 1, la segunda al alumno 2 y así sucesivamente.

Solución inicial sin prioridad:

Iteración	Estado de cambios de alumnos	Lista Tabú	Resultado de la iteración
0	0010000	0,0,0,0,0,0,0	El número de cambios en la solución inicial es: 1

Ilustración 22. Solución inicial generada por Búsqueda Tabú programado, para muestra de datos sin prioridad

El conjunto de iteraciones con el que el método programado encuentra la solución, se presenta en el Apéndice 1.

La solución final encontrada por el heurístico Búsqueda Tabú programado en esta tesis, se presenta en la siguiente ilustración.

Iteración	Estado de cambios de alumnos	Lista Tabú	Resultado de la iteración
50	1100110	0,0,0,0,2,0,0	Del conjunto de iteraciones, la mejor hasta la : 50, tiene un valor de: 4 Asignación final : 1100110 El número de cambios final es :4

Ilustración 23. Solución final generada por Búsqueda Tabú, para muestra de datos sin prioridad

2.- Segundo caso, alumnos con prioridad. La solución óptima es de cuatro cambios, siendo la combinación de alumnos 1,2,4,5 .

a) Muestra de datos de alumnos para la ejecución del heurístico Búsqueda Tabú programado, utilizando prioridad de asignación.

No. Consecutivo de alumno	Escuela de origen	Escuela solicitada	Calificación	Prioridad
1	2	1	22	1
2	1	3	32	1
3	4	5	36	1
4	1	2	40	2
5	3	5	22	1
6	1	2	40	1
7	6	5	31	1

Ilustración 24 muestra de datos de alumnos para la ejecución de Búsqueda Tabú, con prioridad de asignación

b) Muestra de datos de escuelas para la ejecución del heurístico Búsqueda Tabú programado, utilizando prioridad de asignación.

No. de escuela	Lugares disponibles
1	0
2	0
3	0
4	0
5	1
6	0

Ilustración 25. muestra de datos de escuelas para la ejecución de Búsqueda Tabú, con prioridad de asignación.

Solución inicial con prioridad generada por Búsqueda Tabú es la siguiente:

Iteración	Estado de cambios de alumnos	Lista Tabú	Resultado de la iteración
0	0010000	0,0,0,0,0,0,0	El número de cambios en la solución inicial es: 1

Ilustración 26. Solución inicial generada por Búsqueda Tabú programado, para muestra de datos con prioridad

El conjunto de iteraciones realizadas por el heurístico Búsqueda Tabú programado en esta tesis, se presentan en el apéndice 2.

La solución final encontrada por el heurístico Búsqueda Tabú programado en esta tesis, se presenta en la siguiente ilustración.

Iteración	Estado de cambios de alumnos	Lista Tabú	Resultado de la iteración
50	1101100	0,0,0,0,2,0,0	Del conjunto de iteraciones, la mejor hasta la : 50, tiene un valor de: 5 Asignación final : 1101100 El número de cambios final es :4

Ilustración 27. Solución final generada por Búsqueda Tabú programado, para muestra de datos con prioridad

6.2 RESULTADOS GLOBALES

1) El método tradicional obtiene 5,554 cambios.

2) La variante del método tradicional obtiene 5,673 cambios.

3) En una computadora Pentium III con 128 Mb. en RAM, la ejecución del Búsqueda Tabú se realiza en aproximadamente 30 minutos y obtiene hasta 6,000 cambios con los siguientes números de iteraciones.

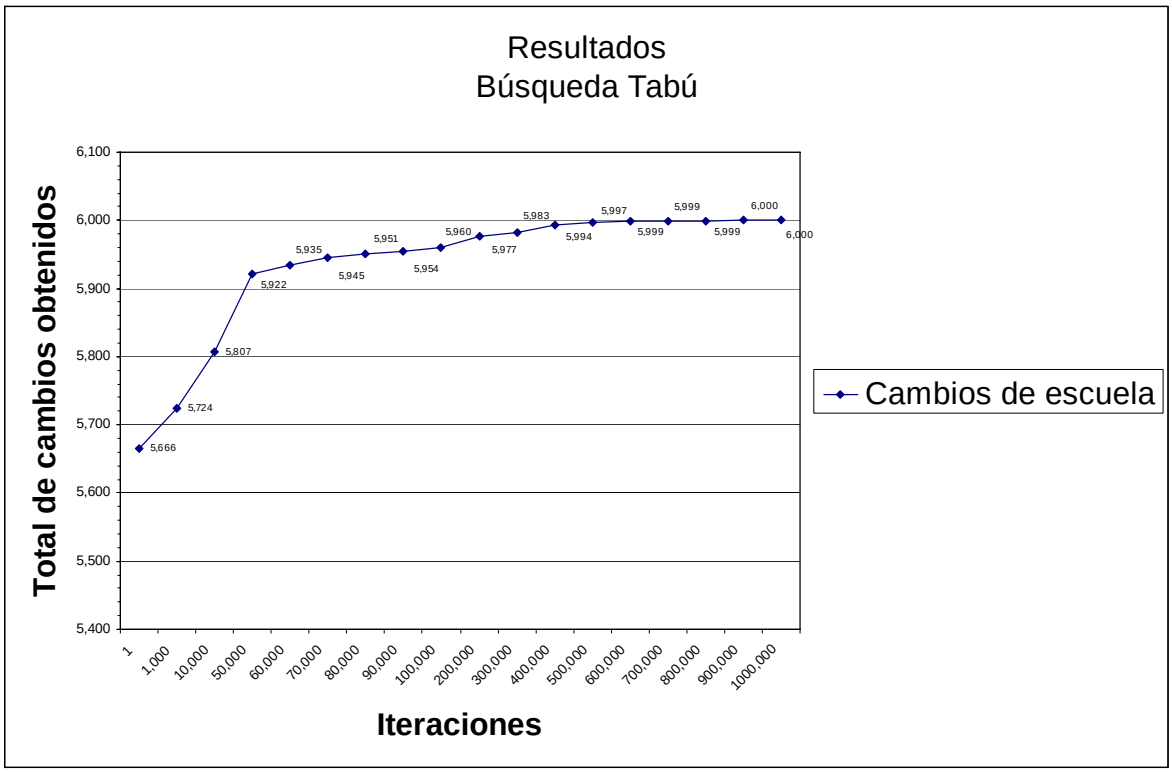


Ilustración 28. Resultados globales

En la gráfica anterior se presenta la evolución de la función objetivo. Aún cuando el total de cambios mostrado tiene un comportamiento ascendente, hacia el interior de la ejecución del método se obtienen oscilaciones para salir de óptimos locales.

6.3 CONCLUSIONES

El problema de cambios de escuela puede modelarse como un sistema de programación entera binaria.

Con una muestra pequeña, al contrastarse los resultados exactos y los obtenidos por Búsqueda Tabú, se encuentra, que para ambos casos con y sin prioridad, el método aquí programado, encuentra las soluciones óptimas.

Las restricciones asignadas a cada alumno reducen considerablemente la oportunidad de mejorar el valor de la función objetivo, ya que para algunos casos donde existen cambios de alumnos con la máxima prioridad (7), se requiere al menos siete cambios de alumnos con prioridades mínimas (1) para igualar el valor de la función objetivo.

La asignación de la misma prioridad a todos los alumnos, permite que Búsqueda Tabú libere ciclos o abrazos de manera más eficiente, lo que permite sugerir la reducción del rango de valores entre prioridades, en aras de obtener mejores resultados.

En situaciones donde es imperante la realización de cambios con mayor prioridad, se puede utilizar un esquema híbrido para la solución, es decir particionar la base de datos y aplicar a cada partición el método seleccionado de cambios (tradicional, variante y BT).

La solución presentada le brinda al responsable del proceso de cambios de escuela, la posibilidad de obtener varios escenarios a partir de la resolución de

los diferentes modelos planteados, lo que le permite tomar decisiones bajo certeza.

Se proporciona a la Dirección de Planeación Educativa, una valiosa herramienta para atender de manera más eficiente a la población escolar que solicita cambios de escuela.

Se comprueba la utilidad del uso de métodos heurísticos para la solución de problemas de naturaleza combinatoria, y se explora un nuevo paradigma de algoritmos de solución dentro de las actividades de planeación y programación en el sector público.

Apéndice 1.

Conjunto de iteraciones realizadas por el programa de Búsqueda Tabú, para la solución del problema de asignación, utilizando muestra de datos sin prioridad. La columna de iteración presenta un asterisco en los casos donde el programa encontró una solución óptima.

Iteraciones realizadas por el programa que implementa el heurístico Búsqueda Tabú en la solución de cambios sin utilizar prioridades de asignación.

Iteración	Lista Tabú Al inicio de la iteración	Estado de cambios de alumnos	Lista Tabú Al final de la iteración	Resultado de la iteración
1	0,0,0,0,0,0,0	0000000	0,0,1,0,0,0,0	El alumno No. 3 entró en lista tabú, un tiempo de: 1 Del conjunto de iteraciones, la mejor hasta la : 1, tiene un valor de: 0
2	0,0,0,0,0,0,0	0000100	0,0,0,0,6,0,0	El alumno No. 5 entró en lista tabú, un tiempo de: 6 Del conjunto de iteraciones, la mejor hasta la : 2, tiene un valor de: 1
3	0,0,0,0,5,0,0	0100100	0,4,0,0,5,0,0	Se intentó cambiar al alumno 5 pero éste está en lista tabú : 5 El alumno No. 2 entró en lista tabú, un tiempo de: 4 Del conjunto de iteraciones, la mejor hasta la : 3, tiene un valor de: 2

Iteración	Lista Tabú Al inicio de la iteración	Estado de cambios de alumnos	Lista Tabú Al final de la iteración	Resultado de la iteración
4	0,3,0,0,4,0,0	1100100	7,3,0,0,4,0,0	El alumno No. 1 entró en lista tabú, un tiempo de: 7 Del conjunto de iteraciones, la mejor hasta la : 4, tiene un valor de: 3
5*	6,2,0,0,3,0,0	1100110	6,2,0,0,3,7,0	El alumno No. 6 entró en lista tabú, un tiempo de: 7 Del conjunto de iteraciones, la mejor hasta la : 5, tiene un valor de: 4
6	5,1,0,0,2,6,0	1100110	5,1,0,0,2,6,0	Se intentó cambiar al alumno 6 pero éste está en lista tabú : 6 Se intentó cambiar al alumno 2 pero éste está en lista tabú : 1 Del conjunto de iteraciones, la mejor hasta la : 6, tiene un valor de: 4
7	4,0,0,0,1,5,0	1100110	4,0,0,0,1,5,0	Se intentó cambiar al alumno 6 pero éste está en lista tabú : 5 Del conjunto de

Iteración	Lista Tabú Al inicio de la iteración	Estado de cambios de alumnos	Lista Tabú Al final de la iteración	Resultado de la iteración
				iteraciones, la mejor hasta la : 7, tiene un valor de: 4
8	3,0,0,0,0,4,0	1000110	3,3,0,0,0,4,0	Se intentó cambiar al alumno 6 pero éste está en lista tabú : 4 El alumno No. 2 entró en lista tabú, un tiempo de: 3 Del conjunto de iteraciones, la mejor hasta la : 8, tiene un valor de: 4
9	2,2,0,0,0,3,0	1000010	2,2,0,0,8,3,0	Se intentó cambiar al alumno 2 pero éste está en lista tabú : 2 El alumno No. 5 entró en lista tabú, un tiempo de: 8 Del conjunto de iteraciones, la mejor hasta la : 9, tiene un valor de: 4
10	1,1,0,0,7,2,0	1010010	1,1,3,0,7,2,0	Se intentó cambiar al alumno 5 pero éste está en lista tabú : 7 El alumno No. 3 entró en lista tabú, un tiempo

Iteración	Lista Tabú Al inicio de la iteración	Estado de cambios de alumnos	Lista Tabú Al final de la iteración	Resultado de la iteración
				de: 3 Del conjunto de iteraciones, la mejor hasta la : 10, tiene un valor de: 4
11	0,0,2,0,6,1,0	1010010	0,0,2,0,6,1,0	Se intentó cambiar al alumno 3 pero éste está en lista tabú : 2 Del conjunto de iteraciones, la mejor hasta la : 11, tiene un valor de: 4
12	0,0,1,0,5,0,0	1010010	0,0,1,0,5,0,0	Se intentó cambiar al alumno 3 pero éste está en lista tabú : 1 Del conjunto de iteraciones, la mejor hasta la : 12, tiene un valor de: 4
13	0,0,0,0,4,0,0	1010010	0,0,0,0,4,0,0	Del conjunto de iteraciones, la mejor hasta la : 13, tiene un valor de: 4
14	0,0,0,0,3,0,0	1000010	0,0,7,0,3,0,0	El alumno No. 3 entró en lista tabú, un tiempo de: 7

Iteración	Lista Tabú Al inicio de la iteración	Estado de cambios de alumnos	Lista Tabú Al final de la iteración	Resultado de la iteración
				Del conjunto de iteraciones, la mejor hasta la : 14, tiene un valor de: 4
15	0,0,6,0,2,0,0	1000011	0,0,6,0,2,0,2	Se intentó cambiar al alumno 3 pero éste está en lista tabú : 6 Se intentó cambiar al alumno 5 pero éste está en lista tabú : 2 El alumno No. 7 entró en lista tabú, un tiempo de: 2 Del conjunto de iteraciones, la mejor hasta la : 15, tiene un valor de: 4
16	0,0,5,0,1,0,1	1000011	0,0,5,0,1,0,1	Se intentó cambiar al alumno 7 pero éste está en lista tabú : 1 iteraciones, la mejor hasta la : 16, tiene un valor de: 4
17	0,0,4,0,0,0,0	1000011	0,0,4,0,0,0,0	Del conjunto de

Iteración	Lista Tabú Al inicio de la iteración	Estado de cambios de alumnos	Lista Tabú Al final de la iteración	Resultado de la iteración
				iteraciones, la mejor hasta la : 17, tiene un valor de: 4
18	0,0,3,0,0,0,0	1000010	0,0,3,0,0,0,1	El alumno No. 7 entró en lista tabú, un tiempo de: 1 Del conjunto de iteraciones, la mejor hasta la : 18, tiene un valor de: 4
19	0,0,2,0,0,0,0	1000110	0,0,2,0,8,0,0	Se intentó cambiar al alumno 3 pero éste está en lista tabú : 2 El alumno No. 5 entró en lista tabú, un tiempo de: 8 Del conjunto de iteraciones, la mejor hasta la : 19, tiene un valor de: 4
20*	0,0,1,0,7,0,0	1100110	0,6,1,0,7,0,0	El alumno No. 2 entró en lista tabú, un tiempo de: 6 Del conjunto de iteraciones, la mejor hasta la : 20, tiene un valor de: 4

Iteración	Lista Tabú Al inicio de la iteración	Estado de cambios de alumnos	Lista Tabú Al final de la iteración	Resultado de la iteración
21	0,5,0,0,6,0,0	1100100	0,5,0,0,6,1,0	Se intentó cambiar al alumno 2 pero éste está en lista tabú : 5 El alumno No. 6 entró en lista tabú, un tiempo de: 1 Del conjunto de iteraciones, la mejor hasta la : 21, tiene un valor de: 4
22*	0,4,0,0,5,0,0	1101100	0,4,0,3,5,0,0	El alumno No. 4 entró en lista tabú, un tiempo de: 3 Del conjunto de iteraciones, la mejor hasta la : 22, tiene un valor de: 4
23	0,3,0,2,4,0,0	1101100	0,3,0,2,4,0,0	Se intentó cambiar al alumno 2 pero éste está en lista tabú : 3 Se intentó cambiar al alumno 4 pero éste está en lista tabú : 2 Del conjunto de iteraciones, la mejor hasta la : 23, tiene un valor de: 4
24	0,2,0,1,3,0,0	1101100	0,2,0,1,3,0,0	Se intentó cambiar al alumno 2 pero éste está

Iteración	Lista Tabú Al inicio de la iteración	Estado de cambios de alumnos	Lista Tabú Al final de la iteración	Resultado de la iteración
				en lista tabú : 2 Se intentó cambiar al alumno 4 pero éste está en lista tabú : 1 Del conjunto de iteraciones, la mejor hasta la : 24, tiene un valor de: 4
25	0,1,0,0,2,0,0	1101100	0,1,0,0,2,0,0	Se intentó cambiar al alumno 2 pero éste está en lista tabú : 1 Del conjunto de iteraciones, la mejor hasta la : 25, tiene un valor de: 4
26	0,0,0,0,1,0,0	1100100	0,0,0,6,1,0,0	El alumno No. 4 entró en lista tabú, un tiempo de: 6 Del conjunto de iteraciones, la mejor hasta la : 26, tiene un valor de: 4
27*	0,0,0,5,0,0,0	1100110	0,0,0,5,0,2,0	Se intentó cambiar al alumno 4 pero éste está en lista tabú : 5 El alumno No. 6 entró en lista tabú, un tiempo de: 2 Del conjunto de

Iteración	Lista Tabú Al inicio de la iteración	Estado de cambios de alumnos	Lista Tabú Al final de la iteración	Resultado de la iteración
				iteraciones, la mejor hasta la : 27, tiene un valor de: 4
28	0,0,0,4,0,1,0	1000110	0,1,0,4,0,1,0	Se intentó cambiar al alumno 6 pero éste está en lista tabú : 1 El alumno No. 2 entró en lista tabú, un tiempo de: 1 Del conjunto de iteraciones, la mejor hasta la : 28, tiene un valor de: 4
29	0,0,0,3,0,0,0	1000010	0,0,0,3,6,0,0	El alumno No. 5 entró en lista tabú, un tiempo de: 6 Del conjunto de iteraciones, la mejor hasta la : 29, tiene un valor de: 4
30	0,0,0,2,5,0,0	1000011	0,0,0,2,5,0,3	Se intentó cambiar al alumno 5 pero éste está en lista tabú : 5 El alumno No. 7 entró en lista tabú, un tiempo de: 3 Estado de cambios

Iteración	Lista Tabú Al inicio de la iteración	Estado de cambios de alumnos	Lista Tabú Al final de la iteración	Resultado de la iteración
				Del conjunto de iteraciones, la mejor hasta la : 30, tiene un valor de: 4

Ilustración 29. Iteraciones realizadas por el heurístico Búsqueda Tabú programado en esta tesis, durante la solución del problema de cambios sin prioridad de asignación.

Apéndice 2.

Conjunto de iteraciones realizadas por el programa de Búsqueda Tabú, para la solución del problema de asignación, utilizando muestra de datos con prioridad. La columna de iteración presenta un asterisco en los casos donde el programa encontró una solución óptima.

Iteraciones realizadas por el programa que implementa el heurístico Búsqueda Tabú en la solución de cambios utilizando prioridades de asignación.

Iteración	Lista Tabú Al inicio de la iteración	Estado de cambios de alumnos	Lista Tabú Al final de la iteración	Resultado de la iteración
1	0000000	0000000	0,0,1,0,0,0,0	El alumno No. 3 entró en lista tabú, un tiempo de: 1 Del conjunto de iteraciones, la mejor hasta la : 1, tiene un valor de: 0
2	0,0,0,0,0,0,0	0000100	0,0,0,0,6,0,0	El alumno No. 5 entró en lista tabú, un tiempo de: 6 Del conjunto de iteraciones, la mejor hasta la : 2, tiene un valor de: 1
3	0,0,0,0,5,0,0	0100100	0,4,0,0,5,0,0	Se intentó cambiar al alumno 5 pero éste está en lista tabú : 5 El alumno No. 2 entró en lista tabú, un tiempo de: 4 Del conjunto de iteraciones, la mejor hasta la : 3, tiene un valor de: 2
4	0,3,0,0,4,0,0	1100100	7,3,0,0,4,0,0	El alumno No. 1 entró en

Iteración	Lista Tabú Al inicio de la iteración	Estado de cambios de alumnos	Lista Tabú Al final de la iteración	Resultado de la iteración
				lista tabú, un tiempo de: 7 Del conjunto de iteraciones, la mejor hasta la : 4, tiene un valor de: 3
5*	6,2,0,0,3,0,0	1101100	6,2,0,7,3,0,0	El alumno No. 4 entró en lista tabú, un tiempo de: 7 Del conjunto de iteraciones, la mejor hasta la : 5, tiene un valor de: 5
6	5,1,0,6,2,0,0	1101100	5,1,0,6,2,0,0	Se intentó cambiar al alumno 2 pero éste está en lista tabú : 1 Se intentó cambiar al alumno 4 pero éste está en lista tabú : 6 Del conjunto de iteraciones, la mejor hasta la : 6, tiene un valor de: 5
7	4,0,0,5,1,0,0	1101100	4,0,0,5,1,0,0	Se intentó cambiar al alumno 4 pero éste está en lista tabú : 5 Del conjunto de iteraciones, la mejor hasta la : 7, tiene un valor de: 5

Iteración	Lista Tabú Al inicio de la iteración	Estado de cambios de alumnos	Lista Tabú Al final de la iteración	Resultado de la iteración
8	3,0,0,4,0,0,0	1001100	3,3,0,4,0,0,0	Se intentó cambiar al alumno 4 pero éste está en lista tabú : 4 El alumno No. 2 entró en lista tabú, un tiempo de: 3 Del conjunto de iteraciones, la mejor hasta la : 8, tiene un valor de: 5
9	2,2,0,3,0,0,0	1001000	2,2,0,3,8,0,0	Se intentó cambiar al alumno 2 pero éste está en lista tabú : 2 El alumno No. 5 entró en lista tabú, un tiempo de: 8 Del conjunto de iteraciones, la mejor hasta la : 9, tiene un valor de: 5
10	1,1,0,2,7,0,0	1011000	1,1,3,2,7,0,0	Se intentó cambiar al alumno 5 pero éste está en lista tabú : 7 El alumno No. 3 entró en lista tabú, un tiempo de: 3 Del conjunto de iteraciones, la mejor hasta la : 10, tiene un

Iteración	Lista Tabú Al inicio de la iteración	Estado de cambios de alumnos	Lista Tabú Al final de la iteración	Resultado de la iteración
				valor de: 5
11	0,0,2,1,6,0,0	1011000	0,0,2,1,6,0,0	Se intentó cambiar al alumno 3 pero éste está en lista tabú : 2 Del conjunto de iteraciones, la mejor hasta la : 11, tiene un valor de: 5
12	0,0,1,0,5,0,0	1011000	0,0,1,0,5,0,0	Se intentó cambiar al alumno 3 pero éste está en lista tabú : 1 Del conjunto de iteraciones, la mejor hasta la : 12, tiene un valor de: 5
13	0,0,0,0,4,0,0	1011000	0,0,0,0,4,0,0	Del conjunto de iteraciones, la mejor hasta la : 13, tiene un valor de: 5
14	0,0,0,0,3,0,0	1001000	0,0,7,0,3,0,0	El alumno No. 3 entró en lista tabú, un tiempo de: 7 Del conjunto de iteraciones, la mejor hasta la : 14, tiene un valor de: 5
15	0,0,6,0,2,0,0	1001001	0,0,6,0,2,0,2	Se intentó cambiar al alumno 3 pero éste está en lista tabú : 6

Iteración	Lista Tabú Al inicio de la iteración	Estado de cambios de alumnos	Lista Tabú Al final de la iteración	Resultado de la iteración
				Se intentó cambiar al alumno 5 pero éste está en lista tabú : 2 El alumno No. 7 entró en lista tabú, un tiempo de: 2 Del conjunto de iteraciones, la mejor hasta la : 15, tiene un valor de: 5
16	0,0,5,0,1,0,1	1001001	0,0,5,0,1,0,1	Se intentó cambiar al alumno 7 pero éste está en lista tabú : 1 Del conjunto de iteraciones, la mejor hasta la : 16, tiene un valor de: 5
17	0,0,4,0,0,0,0	1001001	0,0,4,0,0,0,0	Iteración: 17 Del conjunto de iteraciones, la mejor hasta la : 17, tiene un valor de: 5
18	0,0,3,0,0,0,0	1001000	0,0,3,0,0,0,1	El alumno No. 7 entró en lista tabú, un tiempo de: 1 Del conjunto de iteraciones, la mejor hasta la : 18, tiene un valor de: 5

Iteración	Lista Tabú Al inicio de la iteración	Estado de cambios de alumnos	Lista Tabú Al final de la iteración	Resultado de la iteración
19	0,0,2,0,0,0,0	1001100	0,0,2,0,8,0,0	Se intentó cambiar al alumno 3 pero éste está en lista tabú : 2 El alumno No. 5 entró en lista tabú, un tiempo de: 8 Del conjunto de iteraciones, la mejor hasta la : 19, tiene un valor de: 5
20	0,0,1,0,7,0,0	1101100	0,6,1,0,7,0,0	El alumno No. 2 entró en lista tabú, un tiempo de: 6 Del conjunto de iteraciones, la mejor hasta la : 20, tiene un valor de: 5
21	0,5,0,0,6,0,0	1100100	0,5,0,1,6,0,0	Se intentó cambiar al alumno 2 pero éste está en lista tabú : 5 El alumno No. 4 entró en lista tabú, un tiempo de: 1 Del conjunto de iteraciones, la mejor hasta la : 21, tiene un valor de: 5
22*	0,4,0,0,5,0,0	1100110	0,4,0,0,5,3,0	El alumno No. 6 entró en lista tabú, un tiempo

Iteración	Lista Tabú Al inicio de la iteración	Estado de cambios de alumnos	Lista Tabú Al final de la iteración	Resultado de la iteración
				de: 3 Del conjunto de iteraciones, la mejor hasta la : 22, tiene un valor de: 5
23	0,3,0,0,4,2,0	1100110	0,3,0,0,4,2,0	Se intentó cambiar al alumno 6 pero éste está en lista tabú : 2 Se intentó cambiar al alumno 2 pero éste está en lista tabú : 3 Del conjunto de iteraciones, la mejor hasta la : 23, tiene un valor de: 5
24	0,2,0,0,3,1,0	1100110	0,2,0,0,3,1,0	Se intentó cambiar al alumno 2 pero éste está en lista tabú : 2 Se intentó cambiar al alumno 6 pero éste está en lista tabú : 1 Del conjunto de iteraciones, la mejor hasta la : 24, tiene un valor de: 5
25	0,1,0,0,2,0,0	1100110	0,1,0,0,2,0,0	Se intentó cambiar al alumno 2 pero éste está en lista tabú : 1 Del conjunto de

Iteración	Lista Tabú Al inicio de la iteración	Estado de cambios de alumnos	Lista Tabú Al final de la iteración	Resultado de la iteración
				iteraciones, la mejor hasta la : 25, tiene un valor de: 5
26	0,0,0,0,1,0,0	1100100	0,0,0,0,1,6,0	El alumno No. 6 entró en lista tabú, un tiempo de: 6 Del conjunto de iteraciones, la mejor hasta la : 26, tiene un valor de: 5
27*	0,0,0,0,0,5,0	1101100	0,0,0,2,0,5,0	Se intentó cambiar al alumno 6 pero éste está en lista tabú : 5 El alumno No. 4 entró en lista tabú, un tiempo de: 2 Del conjunto de iteraciones, la mejor hasta la : 27, tiene un valor de: 5
28	0,0,0,1,0,4,0	1001100	0,1,0,1,0,4,0	El alumno No. 2 entró en lista tabú, un tiempo de: 1 Del conjunto de iteraciones, la mejor hasta la : 28, tiene un valor de: 5
29	0,0,0,0,0,3,0	1001000	0,0,0,0,6,3,0	El alumno No. 5 entró en

Iteración	Lista Tabú Al inicio de la iteración	Estado de cambios de alumnos	Lista Tabú Al final de la iteración	Resultado de la iteración
				lista tabú, un tiempo de: 6 Del conjunto de iteraciones, la mejor hasta la : 29, tiene un valor de: 5
30	0,0,0,0,5,2,0	1001001	0,0,0,0,5,2,3	Se intentó cambiar al alumno 5 pero éste está en lista tabú : 5 El alumno No. 7 entró en lista tabú, un tiempo de: 3 Del conjunto de iteraciones, la mejor hasta la : 30, tiene un valor de: 5

Ilustración 30. Iteraciones realizadas por el heurístico Búsqueda Tabú programado en esta tesis, durante la solución del problema de cambios con prioridad de asignación.

Apéndice 3.

Código de los programas principales implementados

```
unit uTabu02;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Db, DBTables, StdCtrls, Buttons, Grids, DBGrids, ExtCtrls;


type

//*********************************************************************
 // Clase tPrepTabu preparación de tablas de información
//*********************************************************************
  tPrepTabu = class
    iOfSet : integer; // Valor que será sumado a los OP vespertinos
    iNumAlu,iNumEsc : integer; // Número de alumnos, número de escuelas
    public
    constructor  TPrepTabu(AOfSet : integer);
    procedure pInitableE(tEpsilon, tEscuela : TTable);
    procedure pInitableA(tSoli, tSolFor : TTable);
    procedure pActAluConse(tAlu, tEscu : TTable); // Actualización de op's por op's consecutivos

  end;
//*********************************************************************
 // Clase TTabu método tabú
//*********************************************************************
  TTabu = class
   iAspiracion, iOfSet : integer; // Valor que será sumado a los OP vespertinos
    iNumAlu,iNumEsc : integer; // Número de alumnos, número de escuelas
    public
    procedure pReadAlu(tAlu : TTable;var iaCali,iaOrgn,iaDest,iaPri : array of
                integer);
    procedure pReadEscu(tEscuela : TTable;var  iaCapEsc : array of integer);
    procedure pGeneral1(var iaSolIni,iaOrgn,iaDest,iaNumEstLLegar,iaNumEstSalir,
            iaCapEsc,iaInd : array of integer);
    function fStatusTabu(i,iLogTabu : integer;var iaMovTabú : array of integer)
            : integer;
```

```
  function fCalcula(var iaPri,iaSolIni : array of integer) : integer;
  procedure pBusquedaTabú(iLogTabu : integer; var iaMovTabú,iaMejorSol,iaSolIni,
        iaPri,iaOrgn,iaDest,iaNumEstLLegar,iaNumEstSalir,iaCapEsc,
        iaCapEscM,iaInd : array of integer);
 end;


//*********************************************************************
 // Clase cambio; permite realizar el proceso de asignación y desasignación de cambios
//*********************************************************************
 TCambio = class
  procedure pCambia(tAlu,tEscuela : TTable); // método para realizar los cambios
  procedure pCambiaI(tAlu,tEscuela : TTable); // Método para restablecerlos cambios
 end;

  TformaTabu = class(TForm)
  dEpsilon: TDataSource;
  tepsilon: TTable;
  dEscuela: TDataSource;
  tEscuela: TTable;
  tCamsec: TTable;
  tCamSol: TTable;
  dCamSec: TDataSource;
  dCamSol: TDataSource;
  dBitacora: TDataSource;
  tBitacora: TTable;
  pTitle: TPanel;
  pCliente: TPanel;
  DBGrid4: TDBGrid;
  DBGrid1: TDBGrid;
  DBGrid2: TDBGrid;
  pMenu: TPanel;
  tbbAsiTabu: TBitBtn;
  tbbAsiSaid: TBitBtn;
  BitBtn1: TBitBtn;
  tbbDesa: TBitBtn;
  tbbSalir: TBitBtn;
  procedure tbbAsiTabuClick(Sender: TObject);
```

```
   procedure tbbAsiSaidClick(Sender: TObject);
   procedure tbbDesaClick(Sender: TObject);
  private
   { Private declarations }
  public
   { Public declarations }
  end;

var
  formaTabu: TformaTabu;

implementation

{$R *.DFM}

//*********************************************************************
// Implementación de los métodos de la clase TPRrepTabu
//*********************************************************************

//***********************************
//Constructor de la clase que prepara los datos para b. Tabu
//***********************************
constructor  TPrepTabu.TPrepTabu(AOfSet : integer);
begin
  //ofset para escuelas técnicas, telesecundarias y anexa a la normal
  iOfSet := AOfSet;
end;

//*********************************************************************
// Método para inicialización de escuelas
//*********************************************************************
procedure  TPrepTabu.pInitableE(tEpsilon, tEscuela : TTable);
 var
 a:array of array of integer;
 begin
   tEscuela.active:=FALSE;
   tEscuela.Exclusive:=TRUE;
   TEscuela.active:=TRUE;
```

```
if (tEscuela.RecordCount > 0) then
  tEscuela.EmptyTable;

with tEpsilon do
 begin
  try
   First;
   iNumEsc := 1;
   while not Eof do
     begin
      // turno matutino
      tEscuela.Append;
      tEscuela.FieldByName('op').value := fieldbyname('op').value;
      tEscuela.FieldByName('turno').value := fieldbyname('turno').value;
      tEscuela.FieldByName('opconsecutivo').value := iNumEsc;
      tEscuela.FieldByName('lugDis').value := fieldbyname('lugdis').value;
      tEscuela.FieldByName('auldis').value := fieldbyname('auldis').value;
      tEscuela.FieldByName('capaul').value := fieldbyname('capaul').value;
      tEscuela.FieldByName('ocumat').value := fieldbyname('ocumat').value;
      tEscuela.FieldByName('estimrep').value := fieldbyname('estimrep').value;
     // tEscuela.FieldByName('estrep2').value := fieldbyname('estrep2').value;
     // tEscuela.FieldByName('lugDisves').value := fieldbyname('lugdisves').value;
     // tEscuela.FieldByName('auldisves').value := fieldbyname('auldisves').value;
     // tEscuela.FieldByName('capves').value := fieldbyname('capves').value;
     // tEscuela.FieldByName('ocuves').value := fieldbyname('ocuves').value;
      iNumEsc := iNumEsc + 1;
      if (fieldbyname('turno').value = 2) then
        begin
         tEscuela.Append;
         tEscuela.FieldByName('op').value := fieldbyname('op').value + iOfSet;
         tEscuela.FieldByName('turno').value := fieldbyname('turno').value;
         tEscuela.FieldByName('opconsecutivo').value := iNumEsc;
    //     tEscuela.FieldByName('lugDis').value := fieldbyname('lugdis').value;
    //     tEscuela.FieldByName('auldis').value := fieldbyname('auldis').value;
    //     tEscuela.FieldByName('capaul').value := fieldbyname('capaul').value;
    //     tEscuela.FieldByName('ocumat').value := fieldbyname('ocumat').value;
    //       tEscuela.FieldByName('estimrep').value := fieldbyname('estimprep').value;
```

```
            tEscuela.FieldByName('lugDis').value := fieldbyname('lugdisves').value;
            tEscuela.FieldByName('auldis').value := fieldbyname('auldisves').value;
            tEscuela.FieldByName('capaul').value := fieldbyname('capves').value;
            tEscuela.FieldByName('ocumat').value := fieldbyname('ocuves').value;
            tEscuela.FieldByName('estimrep').value := fieldbyname('estrep2').value;
            //tEscuela.FieldByName('estrep2').value := fieldbyname('estrep2').value;
            iNumEsc := iNumEsc + 1;
           end;
         Sig;
         end;
         finally
          begin
            tEscuela.first;
            while not tEscuela.eof do
             begin
               tEscuela.edit;
               tEscuela.FieldByName('opconsecutivo').value := tEscuela.RecNo;
               tEscuela.post;
               tEscuela.sig;
             end;
          end;
    end;
  end;
end;



//*********************************************************************
// Método para inicialización de alumnos
//*********************************************************************

procedure TPrepTabu.pIniTableA(tSoli, tSolFor : TTable);
  begin
    iNumALu := 1;
    tSolFor.active := false;
    tSolFor.Exclusive := true;
    tsolfor.Active := true;
    if(tSolFor.RecordCount > 0) then
```

```
tSolFor.EmptyTable;
with tSoli do
try
 first;
 begin
  while not eof do
   begin
    tSolFor.Append;
    tSolFor.FieldByName('posrel').value := FieldByName('opfolio').value;
    tSolFor.FieldByName('llaasi').value := FieldByName('llaasi').value;
    tSolFor.FieldByName('ampl').value := FieldByName('ampl').value;
    tSolFor.FieldByName('calif').value := FieldByName('calif').value;
    tSolFor.FieldByName('motivo').value := FieldByName('motivo').value;
    tSolFor.FieldByName('opsol').value := FieldByName('opsol').value;
    tSolFor.FieldByName('tursol').value := FieldByName('tursol').value;
    tSolFor.FieldByName('opfolio').value := FieldByName('opfolio').value;
    tSolFor.FieldByName('llaasi_fin').value := 0;
    if (FieldByName('ampl').value = 2) then
     tSolFor.FieldByName('oporifin').value := FieldByName('llaasi').value + iOfSet
    else
     tSolFor.FieldByName('oporifin').value := FieldByName('llaasi').value;
    if (FieldByName('tursol').value = 2) then
     tSolFor.FieldByName('opsolfin').value := FieldByName('opsol').value + iOfSet
    else
     tSolFor.FieldByName('opsolfin').value := FieldByName('opsol').value;
    Sig;
   end;
 end;
finally
 begin
  tSolFor.First;
// inicio C1 2001-Feb-28 00:28 hrs.
// Las siguientes líneas son necesarios si el proceso de asignación fue realizado
// y se ejecuta tabú solo con fines demostrativos
 //   while not tSolFor.Eof do
 //    begin
 //    if (FieldByName('cambio').value = 1) then
 //       begin
```

```
      //        tSolFor.edit;
      //        tSolFor.FieldByName('llaasi_fin').value := tSolFor.FieldByName('opsolfin').value;
      //        tSolFor.Post;
      //       end;
      //    tSolFor.Sig;
      //    end;
      // fin C1
       end;
      end;
  end;
//*********************************************************************
// Método para Actualización de op's por op's consecutivos
//*********************************************************************

 procedure TPrepTabu.pActAluConse(tAlu , tEscu : TTable);
  var
   iOpOrigen,iOpDestino : integer;
  begin
   tEscu.Active;
   tEscu.setkey;
   tAlu.active := false;
   tAlu.Exclusive := true;
   tAlu.Active := true;
   tAlu.first;

   while not tAlu.Eof  do
    begin
      iOpOrigen := tAlu.FieldByName('oporifin').value;
      iOpDestino := tAlu.FieldByName('opsolfin').value;
      if (tEscu.FindKey([iOpOrigen])) then
        begin
         tAlu.Edit;
         tAlu.FieldByName('oriconsecutivo').value := tEscu.FieldByName('opconsecutivo').value;
         tAlu.post
        end
        else
         begin
         tAlu.Edit;
```

```
          tAlu.FieldByName('oriconsecutivo').value := 0;
          tAlu.post
         end;

        if (tEscu.FindKey([iOpDestino])) then
         begin
          tAlu.Edit;
          tAlu.FieldByName('soliconsecutivo').value := tEscu.FieldByName('opconsecutivo').value;
          tAlu.post
         end
         else
         begin
          tAlu.Edit;
          tAlu.FieldByName('soliconsecutivo').value := 0;
          tAlu.post
         end;
        tAlu.sig;
      end;
  end;


//*********************************************************************
//Implementación de los métodos de la clase TTabu
//*********************************************************************

//*********************************************************************
// Método para la lectura de alumnos
//*********************************************************************

   procedure TTabu.pReadAlu(tAlu : TTable;var iaCali,iaOrgn,iaDest,iaPri : array of integer);
    var
    i : integer;
    fiALu : text;
    sAlu : string;
    begin
     tAlu.active := true;
     iNumAlu := tAlu.RecordCount;
     tAlu.First;
```

```
      assign(fiAlu,'alu.txt');
      rewrite(fiAlu);
     //  iNumAlu:= 10;
      iAspiracion := 0;
      for i:= 1 to iNumAlu do
        begin
          iaOrgn[i] := tAlu.fieldbyname('Oriconsecutivo').value;
          iaDest[i] := tAlu.fieldbyname('Soliconsecutivo').value;
          iaCali[i] := tAlu.fieldbyname('calif').value;
          iaPri[i] := tAlu.fieldbyname('PesoFin').value;
       //    tAlu.edit;
       //    tAlu.fieldbyname('PesoFin').value := random(7) + 1;
       //    tAlu.post;
           // mientras no se conoce la prioridad
          iaPri[i] := 1;
            iAspiracion := iAspiracion + iaPri[i];

        //  iaPri[i] := random(7) + 1;

          write(fiAlu,iaOrgn[i]);
          tAlu.Sig;
          write(fiAlu,', ');
          write(fiAlu,iaDest[i]);
          write(fiAlu,', ');
          write(fiAlu,iaCali[i]);
          write(fiAlu,', ');
          writeln(fiAlu,iaPri[i]);
        end;
        showmessage(inttostr(iNumAlu)+'  :  '  +  inttostr(iaOrgn[iNumAlu])+
',,'+inttostr(iaDest[iNumAlu]));
        close(fiAlu);
    end;

//*********************************************************************
// Método para la lectura de escuelas
//*********************************************************************
   procedure TTabu.pReadEscu(tEscuela : TTable;var iaCapEsc : array of integer);
    var
```

```
    i : integer;
    fiSalida : text;
    fiEntrada : text;
  begin
    assign(fiSalida,'Escuela.txt');
    rewrite(fiSalida);
    tEscuela.active;
    iNumEsc := tEscuela.RecordCount;
    tEscuela.First;
    showmessage('Escuelas : ' + inttostr(iNumEsc));
    for i:= 1 to iNumEsc do
      begin
        iaCapEsc[i] := tEscuela.FieldByName('lugdis').value - tEscuela.FieldByName('ocumat').value
                    - tEscuela.FieldByName('estimrep').value;
        tEscuela.sig;
        write(fiSalida,i);
        write(fiSalida,',');
        writeln(fiSalida,iaCapEsc[i]);

      end;
      close(fiSalida);
  end;

//*****************************************************************************
// Método de la clase TTabu; Genera una re-configuración inicial
//*****************************************************************************
  procedure TTabu.pGeneral1(var iaSolIni,iaOrgn,iaDest,
              iaNumEstLLegar,iaNumEstSalir,iaCapEsc,iaInd : array of integer);
    var
      i,j,iCambio,iLLegar,iSalir,iSalir_d,iRanAux : integer;
     // iaInd : array of integer;
      fiData : text;  // para guardar datos de seguimiento
    begin
//      assign(fiData,'general.txt');
//      rewrite(fiData);
      //setlength(iaInd,iNumAlu+1);

    for i:= 1 to iNumALu do
```

```
      begin
        iaInd[i] := i;
      end;
    iRanAux := iNumALu;
    for i:= iNumALu  downto 1 do
      begin
        j:= random(iRanAux) + 1;
        iCambio := iaInd[j];
        iaInd[j] := iaInd[i];
        iaInd[i] := iCambio;
        iRanAux := iRanAux - 1;
      end;
    iRanAux := 0;
    for i := 1 to iNumALu do
     begin
      j:= iaInd[i];
       if(iaSolIni[j]=0)  then
        if (iaCapEsc[iaDest[j]] > 0 ) then
              begin
                iaSolIni[j] := 1;
                iaCapEsc[iaDest[j]] := iaCapEsc[iaDest[j]] - 1;
                iaCapEsc[iaOrgn[j]] := iaCapEsc[iaOrgn[j]] + 1;
                iRanAux := iRanAux + 1;
              end;
    end;
     showmessage('SolIni : ' + inttostr(iRanAux));

  end;

//************************************************************
//Método para validar el estado de la lista tabú
//************************************************************

function TTabu.fStatusTabu(i, iLogTabu : integer;var iaMovTabu :
        array of integer): integer;
  var
    j,iFlag : integer;
```

```
   begin

   // Las líneas a continuación comentadas corresponden a la verificación
// Tradicional de la lista tabú

  //    iFlag := 1;
  //    for j:= 1 to iLogTabu do
  //      begin
  //      if(i = iaMovTabu[j]) then
  //         begin
  //          iFlag := 0;
  //         end;
  //      end;
  //       fStatusTabu := iFlag;

      iFlag:=1;
      if(i = iaMovTabu[j]) then
           begin
            iFlag := 0;
           end;
       fStatusTabu := iFlag;
   end;

//*************************************************************
//Método para calcular el valor de una vecindad
//*************************************************************

   function TTabu.fCalcula(var iaPri,iaSolIni : array of integer)
                       : integer;
    var
     iValor, i : integer;
    begin
      iValor := 0;
      for i := 1 to iNumAlu do
        iValor := iValor + iaPri[i] * iaSolIni[i];
      fCalcula := iValor;
    end;
```

```
//*********************************************************************
//Método que realiza la Búsqueda Tabú
//*********************************************************************

 procedure TTabu.pBusquedaTabu(iLogTabu : integer; var iaMovTabu,
          iaMejorSol,iaSolIni,iaPri,iaOrgn,iaDest,iaNumEstLLegar,
          iaNumEstSalir,iaCapEsc,iaCapEscM,iaInd : array of integer);
  var
   i,iIter,iMaxIter, iAle,iLLegar,iSalir,iSalirD : integer;
   iMaximo,iMejor,iMaxPos,iAspiracion,iCambio : integer;
   iCambioT,iFparcial,iCosto,iCostoAux,iRanAux,j,iTiempoTabu : integer;
   sTiempoIni, sTiempoFin : string;
   fiAsigna : text;
  begin
    iMaxIter := 1000000;
  //  for i := 1 to iNumAlu do
  //    iaMejorSol[i] := iaSolIni[i];

    iFparcial := fCalcula(iaPri,iaSolIni);

    iIter := 0;
    iMejor := 0;

    assign(fiasigna,'Resu.txt');
    rewrite(fiasigna);
    sTiempoIni := TimeToStr(Time);
    showmessage('inicio :'+ sTiempoIni);
    while ( iIter < iMaxIter and iMaximo < iAspiracion) do
      begin //3
        iIter := iIter + 1;
        iCambioT := 0;
        iMaximo := 0;
        iMaxPos := -1;
        iCostoAux := -100;
        iCosto := -100;
        ////////////
       // preparación de índices aleatorios
       for i:= 1 to iNumALu do
```

```
 begin
   iaInd[i] := i;
 end;
iRanAux := iNumALu;
for i:= iNumALu  downto 1 do
 begin
   j:= random(iRanAux) + 1;
    iCambio := iaInd[j];
    iaInd[j] := iaInd[i];
    iaInd[i] := iCambio;
    iRanAux := iRanAux - 1;
 end;
 ////////////
 for i:= 1 to iNumALu do
   begin //4
     j:= iaInd[i];
     iCambio := 0;
     if (iaSolIni[j] = 0) then
       begin //  5
         if (iaCapEsc[iaDest[j]] > 0)  then
           begin  // 6
             iCambio := 1;
             iCostoAux := iCambio * iaPri[j];
           end; //6//
       end  //5//
     else
       begin //7
         if ( 0 < iaCapEsc[iaOrgn[j]])  then
           begin //8
             iCambio := -1;
             iCostoAux := iCambio * iaPri[j];
           end; //8//
       end; //7//

     iAle := 51; //random(100);
     //(iAle > 49) and
  //    if ( fStatusTabu(i,iLogTabu,iaMovTabu) = 0) then
     //   showmessage('mov Tabu');
```

```
        //  if ( (iAle > 49) and (iCostoAux >= iCosto) and (iCambio <> 0) and
        //          (fStatusTabu(j,iLogTabu,iaMovTabu) = 1)) then
       // iTiempoTabu := iIter + random(15)+5;
        if ( (iAle > 49) and (iCostoAux >= iCosto) and (iCambio <> 0) and
                (iIter > iaMovTabu[j]) ) then
            begin
              iCosto := iCostoAux;
              iMaximo := iFParcial + iCosto;
              iMaxPos := j;
             iCambioT := iCambio;
              if iMaximo > iMejor then
                begin
                  break;
                end;
            end;
         end;//4//
      if (iCambioT <> 0  ) then
      begin
       if ((iCambioT = 1) ) then //15
         begin
           iaCapEsc[iaOrgn[iMaxPos]] := iaCapEsc[iaOrgn[iMaxPos]] + 1;
           iaCapEsc[iaDest[iMaxPos]] := iaCapEsc[iaDest[iMaxPos]] - 1;
           iaSolIni[iMaxPos] := 1;
         end; //15//
       if (iCambioT = -1) then //16
         begin
           iaCapEsc[iaOrgn[iMaxPos]] := iaCapEsc[iaOrgn[iMaxPos]] - 1;
           iaCapEsc[iaDest[iMaxPos]] := iaCapEsc[iaDest[iMaxPos]] + 1;
           iaSolIni[iMaxPos] := 0;
         end;  //16//
      iFparcial := iFParcial + iCambioT * iaPri[iMaxPos];
       iMaximo := iFparcial;
 //*********************************************************************
 // Actualizar la lista tabú
//**********************************************************************
```

```
    //iaMovTabu[(iIter mod iLogTabu) + 1] := iMaxPos;
      iaMovTabu[iMaxPos] := iIter + (random(16)+5);
    // Actualizar el mejor
    if (iMaximo > iMejor) then
       begin  //17
         iMejor := iMaximo;
         for i:= 1 to iNumAlu do
             iaMejorSol[i] := iaSolIni[i];
         for i:= 1 to iNumEsc do
           iaCapEscM[i] := iaCapEsc[i];
       end; //17//

    //  writeln(fiAsigna,'Iter: ',iIter,'  Maximo: ',iMaximo,' iFParcial: '
    //         ,iFParcial,' iMaxPos: ',iMaxPos);

    iRanAux := 0;
    for i:= 1 to iNumAlu do
      if iaSolIni[i] = 1 then
        iRanAux  := iRanAux + 1;

     writeln(fiAsigna,'Iter: ',iIter,'  Maximo: ',iMaximo,' iFParcial: '
           ,iFParcial,' iMaxPos: ',iMaxPos,' iMejor: ',iMejor);

   end; // if (iCambioT <> 0)
   end; //3//
    iMejor := 0;
    for i:= 1 to iNumAlu do
      begin
        if iaMejorsol[i] = 1 then
         iMejor := iMejor + 1;
//          writeln(fiasigna,iaMejorsol[i]);
      end;
    writeln(fiasigna,'Mejor : ',iMejor );
    iMejor := 0;
    for i:= 1 to iNumAlu do
      if iaSolIni[i] = 1 then
        iMejor  := iMejor + 1;
    writeln(fiasigna,'SolIni : ',iMejor );
```

```
    close(fiasigna);
    sTiempoFin := TimeToStr(Time);
    showmessage('inicio :'+ sTiempoIni+', Fin: '+ sTiempoFin);
  end;

//*********************************************************************
  // método para realizar los cambios
//*********************************************************************
   procedure TCambio.pCambia(tAlu,tEscuela : TTable);
    begin
    end;

  // Método para restablecer los cambios
   procedure TCambio.pCambiaI(tAlu,tEscuela : TTable);
     begin
     end;

//*********************************************************************
// Evento que realiza el llamado a Búsqueda Tabú
//*********************************************************************

procedure TformaTabu.tbbAsiTabuClick(Sender: TObject);
 var
  TIniTableTabu : TPrepTabu;
  TMeTTabu : TTabu;
  k,iLogTabu,iNumAlu,iNumEsc,iNumAux : integer;
  // Datos de los alumnos
  iaCali,iaOrgn,iaDest,iaPri,iaSolIni,iaMejorSol : array of integer;
  // Estudiantes que salen y llegan, capacidad por escuela
  iaNumEstSalir,iaNumEstLLegar,iaCapEsc,iaCapEsc01,iaCapEscM : array of integer;
  iOrigen,iDestino : integer;
  iaMovTabu,iaInd : array of integer;
 begin
  tBitacora.active:= true;
  if  (tBitacora.FieldByName('cambio').value = 1) then
     begin
       showmessage('ya realizó el proceso de asignación,es necesario realizar la desasignación');
     end
```

```
else
  begin
     iLogTabu := 20;
    //Inicialización de tablas de información a partir de epsilon y camsec2k
    //TIniTableTabu := TPrepTabu.TPrepTabu(4000); // ofset para vespertinas
    //TIniTableTabu.pInitableE(tEpsilon,tEscuela);
    //TIniTableTabu.pInitableA(tCamSec,tCamSol);
    //TIniTableTabu.pActAluConse(tCamSol,tEscuela);
   //  setlength(iaMovTabu,iLogTabu+1);   2001jul19
    //Inicialización de arreglos de información de alumnos

    tCamSol.active:= true;
    iNumAlu := tCamSol.RecordCount ;
    iNumAux := iNumAlu + 1;
    setlength(iaSolIni,iNumAux);
    setlength(iaMovTabu,iNumAux);
    setlength(iaMejorSol,iNumAux);
    setlength(iaOrgn,iNumAux);
    setlength(iaDest,iNumAux);
    setlength(iaCali,iNumAux);
    setlength(iaPri,iNumAux);
    setlength(iaInd,iNumAux);

   // Lectura de información de alumnos
   TMeTTabu.pReadAlu(TCamSol,iaCali,iaOrgn,iaDest,iaPri);
   tCamSol.Active:=false;
  //  exit;
   //Inicialización de arreglos de información de escuelas

   tEscuela.active:= true;
   iNumEsc := tEscuela.RecordCount;
   iNumAux := iNumEsc + 1;
   setlength(iaCapEsc,iNumAux);
   setlength(iaCapEscM,iNumAux);
   setlength(iaCapEsc01,iNumAux);
   showmessage('No. de Esc. antres: ' + inttostr(iNumEsc) );

   // Lectura de información de escuelas
```

```
        TMeTTabu.pReadEscu(tEscuela,iaCapEsc);
        //      tEscuela.active:= false;

//       for k:= 1 to iNumEsc do
  //         iaCapEsc01[k] := iaCapEsc[k];

        showmessage('No. de Esc. : ' + inttostr(iNumEsc) );

        for k:= 1 to iNumAlu do
          begin
          //   iaSolIni[k] := 0;
          //   iaMejorSol[k] := 0;
          end;
        for k:= 1 to 1 do
           begin
             tMeTTabu.pGeneral1(iaSolIni,iaOrgn,iaDest,iaNumEstLLegar,
                    iaNumEstSalir,iaCapEsc,iaInd);
                tMeTTabu.pBusquedaTabu(iLogTabu,iaMovTabu,iaMejorSol,iaSolIni,
            iaPri,iaOrgn,iaDest,iaNumEstLLegar,iaNumEstSalir,iaCapEsc,iaCapEscM,iaInd);
           end;
        showmessage('Actualización concluida');

        //actualización de cambios en escuela;
  //     for k:= 1 to iNumAlu do
  //       iaMejorSol[k] := iaSolIni[k];


        tcamsol.Active:= true;
        tEscuela.active := true;
        tEscuela.SetKey;
        tCamsol.first;
        for k := 1 to iNumAlu do //iNumAlu do
            begin
              // if (iaSolIni[k] = 1) then
                 if (iaMejorSol[k] = 1) then
                 begin
                  iOrigen := tCamSol.FieldByName('oporifin').value;
```

```
        iDestino := tCamSol.FieldByName('opsolfin').value;
        tCamsol.edit;
        tCamsol.FieldByName('llaasi_fin').value := iDestino;
        tCamSol.post;
        if (tEscuela.FindKey([iDestino])) then
            begin
              tEscuela.edit;
              tEscuela.FieldByName('ocumat').value := tEscuela.FieldByName('ocumat').value + 1;
              tEscuela.Post;
            end;
          if (tEscuela.FindKey([iOrigen])) then
            begin
              tEscuela.edit;
              tEscuela.FieldByName('ocumat').value := tEscuela.FieldByName('ocumat').value - 1;
              tEscuela.Post;
            end;
        end;
      tCamSol.sig;
    end;


    tBitacora.active:= true;
    tBitacora.Edit;
    tBitacora.FieldByName('cambio').value := 1;
    tBitacora.Post;
    //showmessage(intToStr(iNumCam));
  end;// else de asigación
//  actualización de cambios con búsqueda Tabu
  iaCali := nil;
  iaOrgn := nil;
  iaDest := nil;
  iaPri := nil;
  iaSolIni := nil;
  iaMejorSol := nil;
  iaNumEstSalir := nil;
  iaNumEstLLegar := nil;
  iaCapEsc := nil;
  iaMovTabu := nil;
```

```
        iaInd := nil;
    end;
//*********************************************************************
// método de asignación said
//*********************************************************************
procedure TformaTabu.tbbAsiSaidClick(Sender: TObject);
  var
   iOp,iNumCam,iLugDis : integer;
begin
   tBitacora.active := true;
   tEscuela.Active := true;
   tEscuela.SetKey;
   tCamSol.Active := true;
      if (tBitacora.FieldByName('cambio').value = 1) then
        begin
          // Ya se realizó la asignación
          showmessage('Es necesario realizar la desasignación');
        end
      else
        begin
          // Ejecutar proceso de tradicional de asignación
          tCamSol.first;
          iNumCam := 0;
          iOp := 0;
            while not tCamSol.eof do
             begin
              if (tCamSol.FieldByName('llaasi_fin').value = 0 ) then
               begin
               iOp := tCamSol.FieldByName('opsolfin').value;
               if (tEscuela.FindKey([iOp])) then
                 begin
                  if ((tEscuela.FieldByName('lugdis').value - tEscuela.FieldByName('ocumat').value -
                      tEscuela.FieldByName('estimrep').value ) > 0) then
                    begin
                    // incrementa lugares ocupados en escuela destino
                    tEscuela.edit;
                    tEscuela.FieldByName('ocumat').value := tEscuela.FieldByName('ocumat').value + 1;
                    tEscuela.Post;
```

```
                tCamSol.Edit;
                tCamSol.FieldByName('llaasi_fin').value := iOp;
                iNumCam := iNumCam + 1;
                iOp := tCamSol.fieldByName('oporifin').value;
                //decrementa lugares ocupados en escuela destino
                if(tEscuela.FindKey([iOp])) then
                  begin
                    tEscuela.edit;
                    tEscuela.FieldByName('ocumat').value := tEscuela.FieldByName('ocumat').value - 1;
                    tEscuela.Post;
                  end;
                end;
               end;
              end;
             tCamSol.Sig;
            end;
          tBitacora.Edit;
          tBitacora.FieldByName('cambio').value := 1;
          tBitacora.Post;
          showmessage(intToStr(iNumCam));
        end;
end;
//*********************************************************************
// método de desasignación said
//*********************************************************************
procedure TformaTabu.tbbDesaClick(Sender: TObject);
 var
  iOp,iNumCam : integer;
begin
  tBitacora.active := true;
  tEscuela.Active := true;
  tEscuela.SetKey;
  tCamSol.Active := true;
     if (tBitacora.FieldByName('cambio').value = 0) then
       begin
         // Ya se realizó la desasignación
         showmessage('Ya realizó la desasignación');
       end
```

```
    else
     begin
       // Ejecutar proceso de desasignación
       tCamSol.first;
       iNumCam := 0;
       iOp := 0;
        while not tCamSol.eof do
        begin
          iOp := tCamSol.FieldByName('llaasi_fin').value;
          if iOp > 0 then
           begin
           if (tEscuela.FindKey([iOp])) then
             begin
               tEscuela.edit;
               tEscuela.FieldByName('ocumat').value := tEscuela.FieldByName('ocumat').value - 1;
               tEscuela.Post;
               tCamSol.Edit;
               tCamSol.FieldByName('llaasi_fin').value := 0;
               tCamsol.Post;
               iNumCam := iNumCam + 1;
               iOp := tCamSol.fieldByName('oporifin').value;
               if(tEscuela.FindKey([iOp]))  then
                 begin
                   tEscuela.edit;
                   tEscuela.FieldByName('ocumat').value := tEscuela.FieldByName('ocumat').value + 1;
                   tEscuela.Post;
                 end;
              end;
           end;
          tCamSol.Sig;
        end;
        tBitacora.Edit;
        tBitacora.FieldByName('cambio').value := 0;
        tBitacora.Post;
        showmessage(intToStr(iNumCam));
     end;
end;
end.
```

REFERENCIAS BIBLIOGRÁFICAS

1. F. Glover (1986). “Future paths for integer programming and links to artificial intellingence”. Computers & Ops. Res.; 5, 533-549.

2. F. Glover, M. Laguna (2001). “Tabu Search”, Kluwer Academic Publishers, United States of America.

3. Mitsubishi Electric Research Laboratories. “Human-Guiaded Tabu Search”, http://www.merl.com/projects/GuidedTabu.

4. J.E.Beasley,H.Howells and J.Sonander (2001), “Improving short-term conflict alert”. The Management School, Imperial College, London England.

5. J. Xu, S.Y. Chiu and F. Glover (1998), “Optimizing a Ring-Based Private Line Telecommunication Network Using Tabu Search”, Management Scince.

6. R. Battiti and G. Tecchiolli (1996). “The Reactive Search” (RS) Home Page, http://rtm.science.unitn.it/~battiti/reactive.html.

7. Arturo Fuentes Zenón (1993). “El enfoque de Sistemas en la Solución de Problemas, la Elaboración del Modelo Conceptual”, Facultad de Ingeniería, Universidad Nacional Autónoma de México.

8. C.R. Reeves (1993). “Modern Heuristics Techniques for Combinatorial Problems”, John Wiley & Sons, NY.

9. Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman (1988). “Estructuras de Datos”, Adisson Wesley Iberoamérica.

10. Manuel Laguna (1994). “A Guide to Implementing Tabu Search”, University of Colorado.