Algoritmo de tipo Búsqueda Tabú para un problema de Programación de Horarios Universitarios Vespertinos*

A Tabu Search Algorithm for an Evening University Timetabling Problem

Artículo de Investigación Científica. - Fecha de Recepción: 26 de Marzo de 2013 - Fecha de Aceptación: 15 de Agosto de 2013

Cristian David Oliva San Martín
Ingeniero Civil Industrial. Doctor en Informática. Universidad Católica de la Santísima Concepción. Concepción. Chile. cristian.oliva@ucsc.cl

Gastón Marcelo Ramírez Guzmán
Ingeniero Civil Industrial. Magíster en Gestión de Operaciones y Servicios. Universidad Católica de la Santísima Concepción. Concepción. Chile. gaston.ramirez@magister.ucsc.cl

Para citar este artículo/ To reference this article:
C. Oliva and G. Ramírez, "Algoritmo de tipo búsqueda tabú para un problema de programación de horarios universitarios vespertinos," INGE CUC, vol. 9, no. 2, pp.58–65, 2013.

Resumen: **En este estudio se presenta un modelo de programación no lineal en variables enteras para un problema de programación de horarios universitarios vespertinos y se propone un algoritmo para su solución. El problema consiste en programar asignaturas en un horizonte de planificación considerando profesores, aulas, alumnos, días y un conjunto de restricciones, buscando minimizar la penalidad de no satisfacer los requerimientos de profesores, el número de cambios de sala y el número de periodos libres entre cursos. Se propone un modelo matemático y una implementación de un algoritmo de tipo búsqueda tabú. Para evaluar la efectividad del algoritmo se utilizaron casos de prueba con datos reales del Instituto IPEGE (Chile), en los que el algoritmo es capaz de obtener soluciones factibles en un tiempo razonable. Los parámetros de dicho algoritmo fueron calibrados con los casos de prueba, para posteriormente evaluar su desempeño. Se muestra que este obtuvo mejores soluciones que el método manual.**

Palabras clave: problema de programación de horarios universitarios vespertinos, algoritmo de tipo búsqueda tabú, metaheurísticas.

Abstract: This study presents a model of non-linear integer programming for an evening university timetabling problem and proposes an algorithm for its solution. The problem consists of programming subjects over a planning horizon considering teachers, classrooms, students, days and a set of constraints, trying to minimize the penalty of unsatisfied requirements of teachers, the number of changes of classrooms and the numbers of free periods between courses. A mathematical model is proposed beside the implementation of tabu search algorithm. Test cases are proposed to evaluate the effectiveness of the algorithm, which have real data obtained from IPEGE Institute (Chile), where the algorithm could obtain feasible solutions within a reasonable time. The parameters of the algorithm were calibrated with the test cases to later evaluate its performance, showing that it obtained better solutions than the manual method by an average of 66,5 % and it is about 62,7 % of lower bound calculated.

Keywords: **Evening university timetabling problem, algorithm of tabu search type, metaheuristics.**

* Artículo de investigación derivado del proyecto de investigación titulado: "Estudio y proposición de técnicas hibridas: programación por restricciones y programación lineal para la resolución de problemas de optimización" de la Universidad Católica de la Santísima Concepción. Chile. Fecha de Inicio: 10 de Marzo de 2008. Fecha de Finalización: 30 de Diciembre de 2014.

I. INTRODUCCIÓN

Una gran variedad de problemas de programación de horarios educacionales han sido propuestos en la literatura; estos difieren principalmente de la institución involucrada (universidad o escuela) y las restricciones. Se pueden identificar tres grupos: asignación de horarios escolares, programación de horarios universitarios, asignación de horarios de exámenes [1].

El problema de programación de horarios universitarios vespertinos consiste en programar asignaturas en un horizonte de planificación (generalmente una semana) que se dictan en un periodo académico determinado (año, semestre o trimestre) considerando profesores, aulas, alumnos, días y un conjunto de condiciones y restricciones, de acuerdo con los criterios de cada institución, que determinan la calidad de la programación de horarios y que generalmente varían de una institución a otra. Por este motivo es difícil conseguir una solución que se adapte a todas las instituciones. En [1, 2, 3, 4, 5, 6 y 8] se pueden encontrar algunos estudios entre la gran variedad que existe.

Los métodos utilizados para resolver este tipo de problemas se pueden clasificar en exactos y no exactos [11]. Los primeros consisten en formular un modelo matemático y resolverlo para minimizar o maximizar una función objetivo; en [2 y 7] se presentan problemas resueltos por este método. Los métodos no exactos utilizan procedimientos heurísticos y metaheurísticos, que si bien no encuentran la solución óptima, tienen como ventaja un menor tiempo de cómputo para resolver el problema. Entre los algoritmos de metaheurísticas utilizados para resolver este tipo de problemas se pueden mencionar GRASP [9], recocido simulado [10], algoritmo evolutivo [11, 12] y Tabu Search (búsqueda tabú) [1, 3 y 14], entre otros. En [14] se propone un algoritmo memético basado en búsqueda tabú en el que se combina un algoritmo genético, el cual es utilizado para seleccionar soluciones desde una población de soluciones. El algoritmo tabú penaliza las estructuras de vecindad que no generan mejores soluciones. Los autores mencionan que este algoritmo encuentra algunas de las mejores soluciones de las instancias propuestas en ITC2007.

Este artículo busca resolver el problema de programación de horarios universitarios vespertinos definido en la sección 1, Descripción del problema; el modelo matemático es presentado en la sección 2; en la sección 3, Algoritmo de tipo búsqueda tabú, se presentan los detalles de implementación del algoritmo propuesto; en la sección 4, Experimentos y resultados, se describen los experimentos realizados y los resultados obtenidos; finalmente, en la sección 5, Conclusiones y recomendaciones, se presentan las conclusiones y sugerencias para trabajos futuros.

A. Descripción del problema

El problema de programación de horarios universitarios vespertinos consiste en programar el horario de las asignaturas respetando cada una de las restricciones del problema y buscando satisfacer los requerimientos de horario de los profesores, el número de cambios de sala y el número de periodos libres para cada grupo de alumnos (alumnos que pertenecen a un mismo conjunto de asignaturas). La jornada de clases vespertina está constituida por 5 bloques diarios para dictar las asignaturas, que pueden ser comunes a más de un grupo. Además, las asignaturas tienen preestablecido los profesores que las dictan, la cantidad de periodos consecutivos en los cuales deben dictarse, las aulas y días en que se pueden dictar.

B. Modelo matemático

El modelo matemático que se formula, usando como referencia el estudio [2], contempla todas las restricciones del problema, utilizando un enfoque global. A continuación se definen los conjuntos, parámetros y variables que utiliza el modelo.

1) Definición de conjuntos

C = {1,…, c} es el conjunto de asignaturas por programar.

G = {1,…, g} es el conjunto de grupos de estudiantes. Cada grupo $g \in G$ contiene un conjunto de asignaturas $CG_g \subseteq C$, que no pueden ser programadas en un mismo periodo. Dos grupos distintos pueden tener asignaturas iguales, esto es, sea CG_{g1} y CG_{g2} dos asignaturas pertenecientes a dos grupos, g1 y g2, respectivamente, con g1≠g2 puede ocurrir que $CG_{g1} \cap CG_{g2} \neq \emptyset$.

T = {1,…, t} es el conjunto de periodos de tiempo en que se deben programar las asignaturas en el horizonte de planificación.

D = {1,..., d} es el conjunto de días pertenecientes al horizonte de planificación.

$TDd \subseteq T$ es el conjunto de los periodos que pertenecen al día $d \in D$.

S = {1,…, s} es el conjunto de profesores que deben dictar alguna de las signaturas por programar.

R = {1,…, r} es el conjunto de aulas en las que se puede programar alguna asignatura.

$CS_S \subseteq C$ es el conjunto de asignaturas que dicta el profesor $s \in S$.

$RC_c \subseteq R$ es el conjunto de aulas disponibles para dictar la asignatura $c \in C$.

$TC_c \subseteq T$ es el conjunto de periodos en los que se puede dictar la asignatura $c \in C$.

$TS_S \subseteq T$ es el conjunto de periodos en que el profesor $s \in S$ **no** está disponible para dictar una asignatura.

A partir de los conjuntos definidos anteriormente es posible obtener otros conjuntos necesarios para generar el modelo:

$CRT_{rt} \subseteq C$ es el conjunto de asignaturas que se pueden dictar en el aula $r \in R$ y en el periodo $t \in T$.

$DC_c \subseteq D$ es el conjunto de días en que se puede dictar la asignatura $c \in C$.

$CGD_{gd} \subseteq C$ es el conjunto de asignaturas que se pueden dictar para el grupo $g \in G$ el día $d \in D$.

$CST_{st} \subseteq C$ es el conjunto de asignaturas que dicta el profesor $c \in S$ y factibles de ser dictadas en el periodo $t \in T$.

$TCD_{cd} \subseteq T$ es el conjunto de periodos en que se puede dictar la asignatura $c \in C$ el día $d \in D$.

$CGT_{gt} \subseteq C$ es el conjunto de asignaturas que se pueden dictar en el periodo $t \in T$.

2) Parámetros

n_C el número de periodos para ser programados para cada asignatura $c \in C$.

$HC_C = \{h_1, ..., h_C\}$ es el conjunto que indica el tamaño de cada evento (número de horas o periodos consecutivos en que se dicta una asignatura) en que se divide cada asignatura $c \in C$. Por ejemplo, sea c = 1 y sea $HC_1 = \{3,2,1\}$. Esto significa que la asignatura 1 contiene 6 periodos que deben ser asignados, divididos en 3 eventos: el primero de 3 periodos consecutivos, el segundo de 2 y el tercero de 1.

α es una penalidad por asignar el profesor $s \in S$ para que dicte una asignatura en un periodo en el cual no está disponible, β es una penalidad para el grupo $g \in G$ por cada periodo libre que quede durante un día luego del primer periodo de clases que haya tenido, γ es una penalidad para el grupo $g \in G$ por cada cambio de aula durante el día.

3) Variables de decisión

$X_{crt} = 1$ si la asignatura $c \in C$ es programada en el aula $r \in RC_C$, en el periodo $t \in TC_C$, $X_{crt} = 0$, e.o.c.

$U_{cdh} = 1$ si para la asignatura $c \in C$ se asignan h periodos consecutivos el día $d \in DC_C$, $h \in HC_C$, $U_{cdh} = 0$ e.o.c.

$Y_{gdr} = 1$ si la asignatura $c \in G_g$ el día $d \in DC_C$ tiene asignada la sala $r \in RC_C$, $Y_{gdr} = 0$ e.o.c.

$V_{gdt} = 1$ si alguna asignatura $c \in CG_g$ se dicta en el periodo $t \in TC_C$ el día $d \in DC_C$, $V_{gdt} = 0$ e.o.c.

4) Formulación matemática

$$Min \sum_{s\in S}\sum_{t\in TS_s}\left(\sum_{c\in CS_s}\sum_{r\in RC_c} X_{crt}\right)\times\alpha + \sum_{g\in G}\sum_{d\in D}\left(\sum_{\substack{t3\in TD_d \\ \sum_{t1\in TD_d} V_{gdt1}\le 1 \\ V_{gdt2}>0; V_{gdt3}=0 \\ t1\le t2<t3; t2\in TD_d}} \left(V_{gdt2}+V_{gdt3}\right)\right)\times\beta + \sum_{g\in G}\sum_{d\in D}\sum_{\substack{r\in R \\ \sum_{r\in R} Y_{gdr}>0}} (Y_{gdr}-1)\times\gamma \quad (1)$$

Sujeto a:

$$\sum_{r\in RC_c}\sum_{t\in TC_c} X_{crt} = n_c \quad (2)$$

$$c \in C$$

$$\sum_{c\in CGT_{gt}}\sum_{r\in RC_c} X_{crt} \le 1 \quad (3)$$

$$g \in G; t \in T$$

$$\sum_{c\in CS_s}\sum_{r\in RC_c} X_{crt} \le 1 \quad (4)$$

$$s \in S; t \in TC_c$$

$$\sum_{c\in CRT_{rt}} X_{crt} \le 1 \quad (5)$$

$$r \in R; t \in T$$

$$\sum_{r\in RC_c} X_{crt} \le 1 \quad (6)$$

$$c \in C;\ t \in TC_c$$

$$\sum_{r\in RC_c}\sum_{t\in TCD_{cd}} X_{crt} - \sum_{h\in HC_c}\left(U_{cdh}\times h\right)=0$$

$$h \le \left|\mathrm{TCD_{cd}}\right| \qquad (7)$$

$$c\in C; d\in DC_c$$

$$\sum_{r\in RC_c}\left(X_{crt1} - X_{crt2} + X_{crt3}\right)\le 1$$

$$c\in C;\ \exists\, h\in HC_c>1;\ d\in DC_c;\ \left|TCD_{cd}\right|\ge 3; \qquad (8)$$

$$t_1, t_2, t_3 \in TCD_{cd}; t_1<t_2<t_3;\ \left|TCD_{cd}\right|>h$$

$$X_{cr1t1} + X_{cr2t2}\le 1$$

$$c\in C;\ \left|RC_c\right|>1;\ r_1, r_2\in RC_c;\ r_1\neq r_2;\ d\in DC_c;\ t_1,\ t_2\in TCD_{cd}; t_1<t_2;\ t_2-t_1<h;\ \left|TCD_{cd}\right|\ge h;\ \mathrm{h}>1;\ h\in HC_c \qquad (9)$$

$$\sum_{h\in HC_c} U_{cdh}\le 1$$

$$h\le\left|TCD_{cd}\right|;\ c\in C;\ d\in DC_c \qquad (10)$$

$$\sum_{d\in DC_c} U_{cdh}=1$$

$$h\le\left|TCD_{cd}\right|;\ c\in C;\ h\in HC_c \qquad (11)$$

$$Y_{gdr}\ge\frac{\sum_{c\in CGD_{gd}}\sum_{t\in TCD_{cd}} X_{crt}}{\sum_{c\in CGD_{gd}}\sum_{t\in TCD_{cd}} X_{crt}+1} \qquad (12)$$

$$g\in G; d\in D; r\in R$$

$$Y_{gdr}\le\sum_{c\in CGD_{gd}}\sum_{t\in TCD_{cd}} X_{crt} \qquad (13)$$

$$g\in G; d\in D; r\in R$$

$$V_{gdt}=\sum_{c\in CGD_{gd}}\sum_{r\in RC_c} X_{crt} \qquad (14)$$

$$g\in G; d\in D; t\in TCD_{cd}$$

$$\begin{aligned}
&X_{crt}\in\{0,1\} && c\in C; r\in RC_c;\ t\in TC_c\\
&U_{cdh}\in\{0,1\} && c\in C; d\in {}_{DC_c};\ h\in {}_{HC_c}\\
&Y_{gdr}\in\{0,1\} && g\in G; c\in CG_g; d\in {}_{DC_c}; r\in RC_c\\
&V_{gdt}\in\{0,1\} && g\in G; c\in CG_g; d\in {}_{DC_c}; t\in TC_c
\end{aligned} \qquad (15)$$

5) Restricciones secundarias

La función objetivo (1) se descompone en las siguientes tres partes:

1. Minimizar la penalidad de no satisfacer los requerimientos de horario de los profesores.
2. Minimizar las penalidades por dejar periodos libres para los grupos durante un día, una vez que comenzaron las clases. Por ejemplo, si los días están constituidos de 6 periodos diarios y un grupo tiene clases en los periodos 2, 3 y 5, se penalizarán los periodos 4 y 6 solamente; el periodo 1 no, ya que las clases comenzaron en el periodo 2.
3. Minimizar las penalidades por cambiar de sala a los cursos durante un día.

6) Restricciones primarias

1. Para cada asignatura se debe asignar el número de periodos requeridos.
2. Cada grupo debe tener asignado a lo más una asignatura en un determinado periodo.
3. Cada profesor puede dictar a lo más una asignatura en un determinado periodo.
4. En cada sala se programa a lo más una asignatura por periodo.
5. Cada asignatura se debe dictar a lo más en una sala para un determinado periodo.
6. Las asignaturas son divididas en periodos consecutivos (evento). Por ejemplo: si la asignatura de estadística tiene 5 periodos, puede ser dividida por el usuario en dos eventos de 2 y 3 periodos, respectivamente; los periodos de un evento se deben dictar el mismo día.
7. Los periodos de una asignatura que se programan en un día deben ser consecutivos.
8. Los periodos consecutivos de una asignatura se deben asignar en la misma aula.
9. Para cada asignatura, el número máximo de eventos por asignar para un día determinado es 1.

10. Para cada asignatura un evento debe ser asignado una sola vez.
11. Sirve para asignar valores a las variables Y_{gdr} y satisfacer la tercera restricción secundaria.
12. Cumple el mismo propósito de la restricción anterior.
13. Sirve para asignar valores a las variables V_{gdt} y satisfacer la segunda restricción secundaria.

C. Algoritmo de tipo búsqueda tabú

1) Estructura para representar la solución

La estructura para representar la solución usa listas enlazadas de forma dinámica y permite almacenar los horarios (solución del problema) de los eventos. En la figura 1 se presenta dicha estructura.

Existe un nodo principal que tiene asociada cada una de las aulas de la organización, y estas, a su vez, tienen 6 ramas que representan cada día de la semana con los correspondientes eventos asociados que forman parte de una solución, en la que cada evento tiene el periodo de inicio para el día en que está asociado y su tamaño en periodos. Esta estructura cambia en cada iteración del algoritmo, ya que cada vez se obtiene una solución distinta.

2) Matrices

La matriz val1 [r][d] almacena el valor total de las penalidades para los profesores que dictan clases en el aula *r* el día *d* en un horario que no están disponibles. La matriz val2 [g][d] almacena el valor total de las penalidades por las horas libres que quedan para el grupo *g* el día *d*. La matriz val3 [g][d] almacena el valor total de las penalidades por cambio de sala para el grupo *g* el día *d*. Existe una cuarta matriz, A[g][d][r][t], por medio de la cual se recalculan los valores de las matrices val1, val2 y val3; esta posee el valor 1 si una asignatura del grupo *g* el día *d* usa el aula *r* en el bloque *t*; en caso contrario posee el valor cero.

Ya definidas las matrices para calcular las penalidades, la función objetivo queda dada por la fórmula (16):

$$Min\ \alpha \times \sum_{r \in R} \sum_{d \in D} Val1_{rd} + \beta \times \sum_{g \in G} \sum_{d \in D} Val2_{gd} + \gamma \times \sum_{g \in G} \sum_{d \in D} Val3_{gd} \quad (16)(17)$$

3) Generación del vecindario

Para generar el vecindario lo que se hace es seleccionar un aula y día al azar en los que se puede dictar el evento *x* al que se le va a generar el vecindario; luego se analiza la posibilidad de hacer algunos movimientos del evento *x* en el aula y día escogida (los movimientos se pueden realizar a lo más entre dos eventos), ya sea solo o intercambiando con algún otro evento. Este proceso se repite mientras no queden más eventos por analizar y hasta alcanzar un TV (tamaño del vecindario) menor o igual a *y/100 x E*, donde *y* es una constante que se ajusta en el proceso de calibración de TV y *E* corresponde al número total de eventos del problema.

D. Constructivo de la solución inicial

El constructivo consta de cuatro fases:

1) Primera fase

Se crea un "ranking" de las aulas y días más solicitados por los eventos, luego se establece un "ranking" de estos últimos de acuerdo con la puntuación que asigna la fórmula (17), la cual tiene relación con el número de periodos en los que se puede iniciar un evento en las distintas aulas durante la semana. Los eventos son ordenados en el "ranking" de menor a mayor puntuación en una lista de espera, para luego tratar de asignar los más difíciles primero (los de menor puntuación).

puntuaciónevento = d x a x (b - t - n + 2) (17) corresponde al número de días en que se puede dictar un evento; *a* es el número de aulas en que se puede dic-

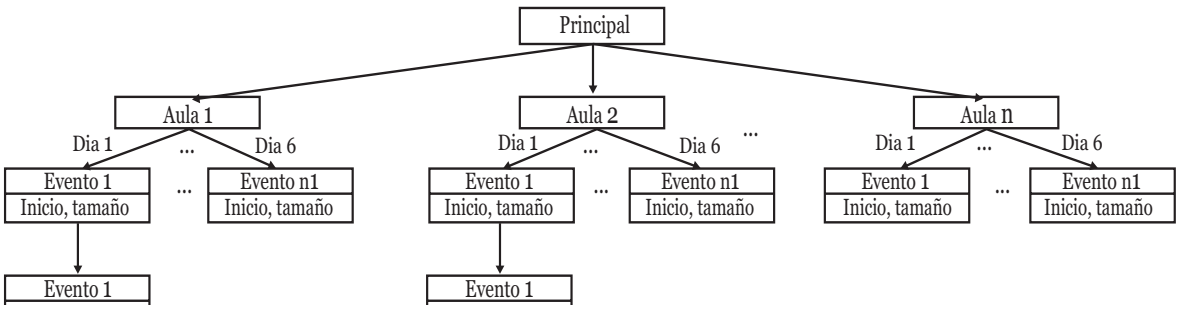

Figura 1. Esquema de la estructura de la solución
Fuente: Elaboración propia.

tar un evento, *b* es el número de bloques por día (lunes a viernes), *t* es el número de periodos que constituyen un evento y *n* es el número de grupos a los cuales pertenece un evento.

2) Segunda fase

Aquí se pretende asignar los eventos de menor a mayor puntuación en el "ranking" (se van asignando los eventos de mayor a menor complejidad) a las aulas y días que tienen menor puntuación ("ranking"), ya que son los menos solicitados. En una primera etapa se trata de asignar los eventos respetando las restricciones secundarias 1 y 2; si no se puede, en una segunda etapa solo se intentan ubicar en cualquier aula y día en que no se viole ninguna restricción primaria.

3) Tercera fase

Si aún quedan elementos en la lista de espera, el constructivo intentará insertarlos de acuerdo con el "ranking" (menor a mayor); para ello se crea un vecindario de los eventos que ocupan aulas y días en que un determinado evento de la lista se quiere agregar. Por cada permutación posible se prueba (si es que se lleva a cabo) la factibilidad de insertar el evento pendiente en la sala *r* y día *d*, donde estaba el evento al cual se le generó el vecindario.

4) Cuarta fase

Si aún quedan elementos pendientes, se duplica la lista de espera escogiendo al azar un evento vecino de cada uno; posteriormente se realiza búsqueda tabú con los pesos anulados; después de *i* iteraciones (0,1 * E) se intenta insertar los elementos pendientes; posteriormente, si la lista de espera es mayor a cero, se realizan *i* iteraciones, para luego tratar de insertar los eventos nuevamente; si todavía quedan eventos en la lista de espera, se repite este proceso desde el comienzo.

5) Lista tabú

La lista tabú usa memoria atributiva, y por cada iteración se almacena el rut del profesor, el tamaño y el o los grupo(s) de cada evento (grupos o cursos a los que pertenece el evento) involucrado en el movimiento con una duración TT (*tabú tenure*, que se ajusta en el proceso de calibración de parámetros) y a su vez se descuenta en una unidad la duración de cada movimiento. Antes de realizar un movimiento se verifica que el par de grupos de atributos de los eventos involucrados no permanezcan en alguno de los nodos (movimientos) de la lista tabú.

6) Criterio de aspiración

Si un movimiento se encuentra en la lista tabú, no se puede realizar, a no ser que permita encontrar una solución mejor que el óptimo global.

7) Intensificación

Una vez que se han hecho NIPI (número de iteraciones previo por intensificar) iteraciones y la solución no mejora, se recupera la mejor solución obtenida hasta el momento y se intensifica la búsqueda es esa zona.

8) Diversificación

Cuando el número de iteraciones sin que la solución mejore es mayor que $z \times NC/NA$, se selecciona un evento al azar y se continua la búsqueda en la zona donde se encuentra este z es una constante por ajustar en el proceso de calibración de parámetros, NC corresponde al número total de cursos y NA es el número total de aulas, por lo tanto, entre más alto es el valor de la razón NC/NA, los eventos están más interrelacionados (el vecindario promedio de los eventos es mayor), y como consecuencia, el NIPD (número de iteraciones que deben transcurrir antes de diversificar) debe ser mayor.

9) Condición de término

Existe una sola condición de término: el tiempo de iteración del algoritmo; este es un parámetro que ingresa el usuario.

E. Experimentos y resultados

1) Recursos empleados y casos de prueba

El algoritmo fue desarrollado con listas enlazadas en C++, en un computador con procesador AMD de 1,8 GHz, 256 MB de memoria RAM y sistema operativo Windows. XP, usando datos reales del Instituto IPEGE (Chile). En la tabla 1 se muestran las características de los casos de prueba:

Tabla I. Casos de prueba

	2009 Semestre II caso 1	2010 Semestre I caso 2	2010 Semestre II Caso 3
Profesores	44	35	31
Aulas	15	14	14
Carreras	13	7	6
Asignaturas	89	72	59
Eventos	117	106	82
Grupos	17	13	9

Fuente: Elaboración propia.

2) Procedimiento de evaluación

- Fijar los parámetros a valores que parezcan razonables. Se definen en conjunto con la administración de IPEGE.
- Realizar 10 iteraciones por cada valor distinto que toma cada uno de los parámetros y luego calcular sus promedios, y fijar los parámetros a valores en que el algoritmo alcanza el mayor rendimiento.
- Comparar los resultados con el método manual y las cotas inferiores.

Los parámetros con los valores por fijar son: T (tiempo de ejecución) a 30 segundos, TV a 0,1*E, NIPD a 26*NC/NA, NIPI a 1000 iteraciones, TT a un valor aleatorio entre 0,01*E y 0,03*E, α a 0, β a 5, γ a 3. α tiene el valor cero, porque no se pudo obtener la disponibilidad de los profesores para los casos de prueba.

3) Resultados

En la tabla 2 se presentan los promedios de la función objetivo para los tres casos luego de realizar 10 iteraciones por cada valor distinto que toma cada uno de los parámetros:

Tabla II. Resultados de los promedio de la función objetivo para cada uno de los parámetros

Tmin Tmax	NIPI	TV	NIPD	T	Caso 1	Caso 2	Caso 3
0,01*E 0,03*E	1000	0,1*E	26*NC/NA	30	106	22	10
0,1*E 0,3*E	1000	0,1*E	26*NC/NA	30	121	21	12
0,05*E 0,2*E	1000	0,1*E	26*NC/NA	30	144	22	9
0,01*E 0,03*E	1000	0,1*E	26*NC/NA	30	106	22	10
0,01*E 0,03*E	2000	0,1*E	26*NC/NA	30	145	21	11
0,01*E 0,03*E	1000	0,05*E	26*NC/NA	30	143	22	10
0,01*E 0,03*E	1000	0,1*E	26*NC/NA	30	106	22	10
0,01*E 0,03*E	1000	0,15*E	26*NC/NA	30	118	22	10
0,01*E 0,03*E	1000	0,1*E	26*NC/NA	30	106	22	10
0,01*E 0,03*E	1000	0,1*E	52*NC/NA	30	114	21	11
0,01*E 0,03*E	1000	0,1*E	78*NC/NA	30	123	22	9
0,01*E 0,03*E	1000	0,1*E	26*NC/NA	30	106	22	10
0,01*E 0,03*E	100	0,1*E	52*NC/NA	60	94	20	10
0,01*E 0,03*E	1000	0,1*E	78*NC/NA	90	88	21	10

Fuente: Elaboración propia.

Los valores que están en negro son los que pueden tomar los parámetros, y de estos, los que se muestran en cursiva son en los cuales el algoritmo alcanza el mayor rendimiento.

En el gráfico de la figura 2 se muestra el valor de la $F.O_i$ (valor de la función objetivo entregada por el constructivo), $F.O_f$ (valor de la función objetivo obtenida por el algoritmo), el método manual y las cotas inferiores con los parámetros ya ajustados. Para calcular estas cotas se identifican algunas situaciones en que es inevitable violar las restricciones secundarias 2 y 3.

Se puede estimar que el servicio al cliente mejoró en promedio en un 66,5% ([[173-88]/173 + [88-21]/88 + [39-10]/39]/3*100) en lo que tiene relación con la programación de horarios con respecto a las soluciones conseguidas por el método manual, por lo tanto se cumplió con el objetivo de este estudio. Para los casos de prueba, el algoritmo está en promedio a lo más en un 62,7 % ([[88-60]/60 + [21-12]/12 + [10-9]/9]/3*100) por debajo de la calidad de la solución óptima.

La solución final mejoró en promedio en un 72 % ([[308-88]/308+[56-21]/56+[55-10]/55]/3*100) respecto a las conseguidas por el constructivo de la solución inicial, con lo que se demuestra que la etapa de mejora de la solución inicial es eficiente.

IV. Conclusiones y Recomendaciones

En este trabajo se presentó un problema de programación de horarios universitarios vespertinos, que es complejo por la gran cantidad de variables y restricciones y lo ajustado de los horarios, por lo cual queda muy poca capacidad ociosa, lo que hace difícil obtener una solución inicial. Para este problema se propone un modelo de programación entera y un algoritmo de tipo búsqueda tabú, con el cual se obtuvieron resultados que mejoran el servicio al cliente en comparación con la solución obtenida con el método manual.

La estructura de tipo árbol con listas enlazadas que almacena la solución mejora la eficiencia del algoritmo, lo cual mejora las operaciones de cálculo y de búsqueda, haciendo que este sea rápido, y alcance en el primer caso alrededor de 3000 iteraciones por segundo. Sería interesante evaluar si usando este tipo de estructuras en algunos problemas en los que los tiempos de cómputo deben ser bajos, se pueden obtener buenas soluciones en el tiempo requerido.

Para la realización de trabajos futuros se hacen las siguientes recomendaciones:

- Generar el vecindario con un opt 3; como los horarios están muy ajustados, probablemente con esta modificación se podría reubicar los eventos que están siendo penalizados.
- Realizar análisis de sensibilidad con un opt 3 para contemplar situaciones que no están presentes al momento de correr el *software*.

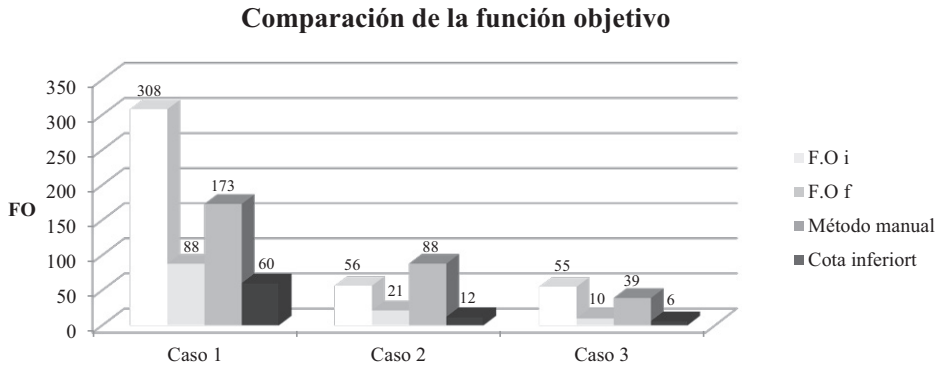

Figura 2. Comparación de los promedios finales de la función objetivo
Fuente: Elaboración propia.

- En la puntuación para generar el "ranking" de los eventos considerar el porcentaje de ocupación semanal del profesor que dicta el evento y la cantidad de asignaturas asociadas al grupo que este último pertenece, para obtener un "ranking" más representativo de la situación.

Referencias

[1] J. Franco, E. Toro y R. Gallego, "Problema de asignación **óptima de salones resuelto con búsqueda t**abú", *Ingeniería y Desarrollo* (Universidad del Norte), vol. 24, pp. 149-175, 2008.

[2] A. Saldaña, C. Oliva, and L. Pradenas, "Models of integer programming for an university Timetabling problem", *Ingeniare* , vol. 15, pp. 245-259, 2007.

[3] R. Alvarez, E. Crespo, and J. Tamarit, "A tabú Search algorithm to schedule university examinations", *QUESTIIO*, vol. 21, pp. 201-215, 1997.

[4] E. Mooney and R. Rardin, "Tabu search for a class of scheduling problems", Annals of Operations Research, vol. 41, pp. 253-278, 1993.

[5] W. Parmenter, E. Mooney, and R. Rardin, "Large scale classroom scheduling", IIE Transactions, vol. 28, pp. 369-378, 1996.

[6] A. Molina, "Algoritmos evolutivos para la resolución de un problema de tipo Timetabling", tesis, Universidad de Valparaíso, Chile, 2007.

[7] R. Hernández, J. Miranda, and P. Rey, "Programación de horarios de clases y asignación de salas para la Facultad de Ingeniería de la Universidad Diego Portales mediante un enfoque de programación entera", *Ingeniería de Sistemas,* vol. 22, pp. 123-143, 2008.

[8] S. Abarca, "Sistema de asignación de horarios de clases", tesis, Universidad Católica de Valparaíso, Chile, 1996.

[9] A. Moura, and R. Scaraficci, *International journal of operational research*, vol. 7, 19, pp. 152-170, 2010.

[10] T. Cura (2007) Istanbul Ticaret Üniversitesi [Online]. Disponible: http://www.iticu.edu.tr/kutuphane/dergi/f12/M00196.pdf.

[11] J. Mejía, "Asignación de horarios de clases universitarias mediante algoritmos evolutivos", tesis, Universidad del Norte, Barranquilla(Colombia), 2008.

[12] P. Flores, E. Brau, J. Monteverde, N. Salazar, J. Figueroa, E. Cadena y C. Lizárraga, "Experimentos con algoritmos genéticos para resolver un problema real de programación Maestros-Horarios-Cursos", *Revista Iberoamericana de Sistemas, Cibernética e Informática*, vol. 1, pp. 42-46, 2004.

[13] ILOG CPLEX 9.0 User's Manual. IBM Co., 2003.

[14] S. Abdullah and H. Turabieh, "On the use of multi-neighborhood structures within a Tabu-based memetic approach to university timetabling problems", *Information Sciences*, Vol 191, pp.146-168, 2012.