Revista de Matemática: Teoría y Aplicaciones
ISSN: 1409-2433
mta.cimpa@ucr.ac.cr
Universidad de Costa Rica
Costa Rica

CHÁVEZ-BOSQUEZ, OSCAR; POZOS-PARRA, PILAR; GÓMEZ-RAMOS, JOSÉ LUIS
BÚSQUEDA TABÚ CON CRITERIO DE ASPIRACIÓN PROBABILÍSTICO APLICADA A LA GENERACIÓN DE HORARIOS ESCOLARES
Revista de Matemática: Teoría y Aplicaciones, vol. 22, núm. 1, enero, 2015, pp. 153-177
Universidad de Costa Rica
San José, Costa Rica

REVISTA DE MATEMÁTICA: TEORÍA Y APLICACIONES 2015 **22**(1) : 153–177
CIMPA – UCR ISSN: 1409-2433 (PRINT), 2215-3373 (ONLINE)

BÚSQUEDA TABÚ CON CRITERIO DE ASPIRACIÓN PROBABILÍSTICO APLICADA A LA GENERACIÓN DE HORARIOS ESCOLARES

TABU SEARCH WITH ASPIRATION CRITERION FOR THE TIMETABLING PROBLEM

OSCAR CHÁVEZ-BOSQUEZ* PILAR POZOS-PARRA†
JOSÉ LUIS GÓMEZ-RAMOS‡

Received: 13/Feb/2014; Revised: 27/Aug/2014;
Accepted: 26/Sep/2014

---

*División Académica de Informática y Sistemas, Universidad Juárez Autónoma de Tabasco. Cunduacán, Tabasco, México. E-Mail: oscar.chavez@ujat.mx
†Misma dirección que/*same address as* O Chávez. E-Mail: pilar.pozos@ujat.mx
‡Misma dirección que/*same address as* O. Chávez. E-Mail: jose.gomez@ujat.mx

Resumen

El criterio de aspiración es un elemento determinante en el funcionamiento del algoritmo de Búsqueda Tabú, siendo el criterio de aspiración por defecto y el criterio de aspiración por objetivo los dos utilizados mayormente en la literatura. En este artículo se propone una variante a estos criterios de aspiración, la cual implementa una función de probabilidad al momento de evaluar un elemento clasificado como tabú que mejore la solución actual: criterio de aspiración probabilístico. El caso de prueba utilizado para evaluar el desempeño del algoritmo de Búsqueda Tabú con Criterio de Aspiración Probabilístico (BT-CAP) propuesto lo constituyen las 20 instancias del problema descrito en *the First International Timetabling Competition*. Se compararon los resultados del BT-CAP contra 2 variantes adicionales del algoritmo de Búsqueda Tabú: Búsqueda Tabú con Criterio de Aspiración por Defecto (BT-CAD) y Búsqueda Tabú con Criterio de Aspiración por Objetivo (BT-CAO). Se aplicó una prueba de Wilcoxon con los resultados obtenidos, y se demostró con 99 % de confiabilidad que el algoritmo BT-CAP obtiene mejores soluciones que las otras dos variantes del algoritmo de Búsqueda Tabú.

Palabras clave: búsqueda tabú; criterio de aspiración; programación de horarios; metaheurística.

Abstract

The aspiration criterion is an imperative element in the Tabu Search, with aspiration-by-default and the aspiration-by-objective the mainly used criteria in the literature. In this paper a new aspiration criterion is proposed in order to implement a probabilistic function when evaluating an element classified as tabu which improves the current solution, the proposal is called Tabu Search with Probabilistic Aspiration Criterion (BT-CAP). The test case used to evaluate the performance of the Probabilistic Aspiration Criterion proposed consists on the 20 instances of the problem described in *the First International Timetabling Competition*. The results are compared with 2 additional variants of the Tabu Search Algorithm: Tabu Search with Default Aspiration Criterion (BT-CAD) and Tabu Search with Objective Aspiration Criterion (BT-CAO). Wilcoxon test was applied to the generated results, and it was proved with 99 % confidence that BT-CAP algorithm gets better solutions than the two other variants of the Tabu Search algorithm.

Keywords: tabu search; aspiration criterion; timetabling; International Timetabling Competition; metaheuristic.

Mathematics Subject Classification: 68T20, 90C59.

1 Introducción

La Búsqueda Tabú es una metaheurística utilizada comúnmente para resolver problemas de optimización combinatoria, y ha sido aplicada eficazmente en la resolución de problemas de generación de horarios escolares [24]. En este contexto, la calidad de los horarios escolares obtenidos se mide por la cantidad de restricciones resueltas, y esta calidad depende, entre otros factores, de la configuración de los parámetros del algoritmo. Se ha demostrado que modificando un parámetro específico del algoritmo se logra mejorar la calidad de los horarios obtenidos, ya sea modificando la lista tabú [2] o las vecindades utilizadas [7]. Sin embargo, uno de los parámetros del algoritmo que no es comúnmente sujeto de modificación es el criterio de aspiración, el cual puede ser personalizado de acuerdo al problema que se desea resolver. El objetivo de este trabajo fue determinar el impacto en la calidad de los horarios al modificar este parámetro, así como medir la eficiencia del algoritmo de Búsqueda Tabú con un criterio de aspiración personalizado. La modificación del algoritmo de Búsqueda Tabú con un criterio de aspiración personalizado permite obtener no solamente horarios de mejor calidad de manera eficiente, sino también mejores soluciones en aquellos problemas de optimización combinatoria donde se aplique el algoritmo de Búsqueda Tabú.

Para evaluar el algoritmo se empleó un estándar que consiste en 20 instancias de un problema de generación de horarios escolares bien conocido: *the First International Timetabling Competition* (ITC) [17]. Casi todos los casos encontrados en la literatura utilizan metaheurísticas para la resolución de este problema [10], y en pocos casos se muestran algoritmos determinísticos que resuelvan efectivamente el problema [6]. Esto demuestra el interés por encontrar nuevas formas de resolver problemas conocidos requiriendo la menor cantidad de recursos posible.

Se seleccionó este problema debido a que cada instancia tiene al menos una solución exacta. Luego es posible medir el grado de eficacia de un algoritmo dependiendo de la proximidad del óptimo global se encuentren sus soluciones. Además, es posible utilizarlo como problema de *benchmark* [10], ya que se pueden comparar los resultados obtenidos por un algoritmo dado contra los resultados obtenidos por los participantes del concurso. Por ejemplo [25], a pesar de no participar en el concurso, se compara el desempeño de su algoritmo al resolver este problema contra los resultados de los concursantes. Cabe mencionar que existen dos actualizaciones del concurso, el ITC-2007 [18] y el ITC-2011 [19]. Sin embargo, estas dos versiones pueden considerarse como una extensión del problema original, así este trabajo se enfoca a la solución de los problemas de la primera versión con la finalidad de contar con un algoritmo competitivo

que pueda extenderse para versiones más complejas de asignación de horarios.

Por otra parte, [14] demuestra que el problema de la generación de horarios universitarios es una aplicación particular de este problema, por lo tanto para resolver otras variantes prácticas del problema basta con ampliar la solución propuesta para el problema del ITC.

1.1 Búsqueda tabú

El *Metaheuristics Network Project* [22] define la Búsqueda Tabú como una metaheurística, ya que incluye en sus propias reglas algunas técnicas heurísticas, su función es dirigir y orientar la búsqueda de otro procedimiento más local de búsqueda. Entre las ventajas de esta estrategia podemos mencionar que se puede implementar como una técnica determinística, no aleatoria [13].

Su versatilidad le hace una técnica muy utilizada para resolver el problema de la generación de horarios [1]. Se tienen casos de estudio planteados donde se compara el resultado de un sistema automatizado contra el modelo manual habitual [20], optimización de calendarios deportivos [3], así como *frameworks* que implementan los procedimientos básicos de Búsqueda Tabú para ser utilizados en diferentes problemas de optimización [9, 16]. Es notable su aplicación en el problema de generación de horarios, y una de las razones es debido a que se trata de una técnica relativamente reciente que aún no se ha explorado por completo.

Los elementos esenciales del algoritmo de búsqueda tabú son [12]:

- Solución inicial: Al tratarse de un algoritmo de optimización, la Búsqueda Tabú requiere iniciar el proceso de exploración con una solución construida previamente.

- Lista tabú: Consiste en la lista de movimientos considerados prohibidos, ya que fueron utilizados durante el proceso de búsqueda para generar una solución. Esta lista corresponde a una cola, en la cual el primer elemento en entrar es el primero en salir. El rol principal de este elemento es prevenir ciclos en la búsqueda.

- Esquema de vecindad: En lugar de explorar todo el espacio de soluciones (como ocurriría en un algoritmo de fuerza bruta) la Búsqueda Tabú realiza una búsqueda en un subconjunto de este espacio, llamado vecindad.

- Criterio de aspiración: Este criterio se define para los movimientos considerados prohibidos, se introduce en la Búsqueda Tabú para determinar cuándo pueden ser reemplazadas o eliminadas las restricciones tabú sobre cierto elemento dentro de la lista tabú.

- Función objetivo: El proceso de optimización involucra el uso de una ecuación que contemple las restricciones a maximizar o minimizar. A esta ecuación se le conoce como función objetivo.

El modelar los elementos de la Búsqueda Tabú consiste en encontrar los valores óptimos para cada uno de estos componentes. A este proceso se le denomina sintonización de parámetros [4].

1.2 Generación de horarios escolares

El proceso de generación de horarios escolares (conocido como *timetabling problem*) consiste en relacionar un conjunto de profesores a un conjunto de materias dentro de un periodo de tiempo fijo, regularmente una semana, satisfaciendo un conjunto de restricciones de diferente tipo. El problema está clasificado como NP-Completo, por lo que no existe un algoritmo determinístico que encuentre la solución al problema en un tiempo razonable [24].

La clase de complejidad NP-Completo es un subconjunto de los problemas de decisión NP tal que todo problema NP se puede convertir en cada uno de los problemas NP-Completos. Se puede decir que esta clase de problemas es de los más difíciles NP y muy probablemente no formen parte de la clase de complejidad P [11]. La razón es que de tenerse una solución polinomial para un problema NP-Completo (también conocidos informalmente como *intratables*), todos los problemas NP tendrían también una solución en tiempo polinomial.

Un algoritmo de fuerza bruta para problemas NP-Completos utiliza tiempo exponencial con respecto al tamaño de la entrada. Se desconoce si hay mejores algoritmos, por lo cual es necesario utilizar diferentes enfoques que permitan obtener una solución adecuada en un tiempo razonable [8].

Al tratarse de un problema NP-Completo, encontrar una solución de forma manual al problema de la generación de horarios escolares requiere usualmente la intervención de una o varias personas durante un cierto periodo de tiempo, y la "solución" encontrada puede no ser tal, ya que puede no cumplir con una o más restricciones.

El término restricciones se define como condiciones que debe satisfacer el horario generado, y se clasifican en dos tipos [21]:

- Restricciones duras: condiciones que deben cumplirse obligatoriamente, la violación de alguna origina un horario no válido.

- Restricciones suaves: condiciones que denotan preferencias del usuario y se desea que se cumplan en la medida de lo posible.

Cada institución educativa genera los horarios de clase de manera diferente, debido a diversas políticas institucionales: tipo de contratación de profesores, horas de asesorías, labores administrativas, entre otras, que hacen variar el número y el tipo de restricciones que se deben satisfacer en la generación del horario.

1.3 *The First International Timetabling Competition*

Este problema de generación de horarios escolares fue diseñado por el *Metaheuristics Network Project* [22] y presentado en el marco del congreso *Practice And Theory in Automated Timetabling* [23]. Fue el primer concurso propuesto en el cual el objetivo fue crear un horario semanal para una universidad. El problema consta de 20 instancias que consisten en:

- Un conjunto de eventos para ser calendarizados en 5 días de 9 horas cada uno,
- Un conjunto de salones en los cuales tomarán lugar los eventos,
- Un conjunto de participantes que asisten a los eventos,
- Un conjunto de características que satisfacen los salones y son requeridos por los eventos.

Cada participante asiste a cierto número de eventos y cada salón tiene un tamaño fijo, que equivale al número máximo de participantes que puede albergar. Un horario factible o válido es aquel en el cual todos los eventos han sido asignados a un periodo de tiempo y a un salón de tal manera que las siguientes restricciones duras se satisfacen [17]:

- Restricción *ITC-rd1*: Sólo un evento se imparte en cada salón en un periodo de tiempo,
- Restricción *ITC-rd2*: El salón es lo suficientemente grande para albergar a todos los participantes y satisface todas las características requeridas por el evento,
- Restricción *ITC-rd3*: Ningún participante asiste a más de un evento durante un mismo periodo de tiempo.

Adicionalmente, un horario candidato es penalizado por cada violación de las siguientes restricciones suaves:

- Restricción *ITC-rs1*: Un participante asiste a un evento en el último periodo del día,

- Restricción *ITC-rs2*: Un participante asiste a más de dos eventos consecutivos, por lo que tres eventos consecutivos equivalen a una violación, cuatro eventos equivalen a dos, y así sucesivamente. Los eventos al final del día, seguidos de eventos al inicio del día siguiente, no se cuentan como consecutivos,

- Restricción *ITC-rs3*: Un participante tiene un único evento en el día. Se contabilizan dos violaciones si un participante tiene dos días con un solo evento.

El objetivo del concurso es producir horarios de clase válidos, en los cuales se deben satisfacer todas las restricciones duras y minimizar el número de restricciones suaves que no se cumplen. Este problema puede servir como cota de referencia internacional, ya que las 20 instancias del problema tienen al menos una solución óptima conocida y los 20 archivos correspondientes a cada instancia están disponibles en línea para que cualquier investigador interesado en el tema pueda tratar de resolverlo [17]. Así mismo, los organizadores del concurso proporcionan un programa de *benchmark* que al ser ejecutado en un equipo de cómputo proporciona el tiempo de ejecución, en segundos, que será utilizado como límite para el algoritmo participante.

2 Búsqueda tabú con criterio de aspiración probabilístico

La Búsqueda Tabú es una metaheurística que guía a un procedimiento de búsqueda local, encargándose de evitar que dicho procedimiento caiga en óptimos locales, al prohibir o penalizar ciertos movimientos durante algún número de iteraciones [12].

El propósito de clasificar ciertos movimientos como tabú se realiza a través de la lista tabú del algoritmo, con lo cual es posible aceptar peores soluciones que la actual con el fin de no quedar "entrampada" en un óptimo local.

El criterio de aspiración en el algoritmo está íntimamente ligado a la lista tabú, ya que define la forma en la cual se reemplaza o elimina el estatus tabú de un movimiento en particular. También llamado convenientemente "olvido estratégico", en la gran mayoría de aplicaciones encontradas en la literatura se utilizan alguno de los siguientes criterios de aspiración:

1. Criterio de aspiración por defecto: Si todos los movimientos disponibles se encuentran con estatus tabú, entonces el movimiento "menos tabú", o el movimiento más "viejo" en la lista tabú, es seleccionado.

2. Criterio de aspiración por objetivo: Un movimiento tabú es admitido si el movimento produce una solución mejor a la mejor solución obtenida hasta el momento. Este suele ser el criterio de aspiración mayormente utilizado.

Con base en el uso y aplicación del criterio de aspiración en el algoritmo de Búsqueda Tabú, se propone un criterio de aspiración probabilístico basado en el valor obtenido en la función objetivo en cada iteración: Búsqueda Tabú con Criterio de Aspiración Probabilístico (BT-CAP) [5].

En este criterio propuesto, un movimiento que conduce a una mejor solución que la mejor solución obtenida hasta ahora, es aceptado sin importar su estatus tabú de acuerdo con la función de probabilidad mostrada en la Ecuación (1). Esta función dirige el proceso de búsqueda al aceptar con mayor probabilidad un movimiento tabú si la exploración se encuentra lejos de la región factible, pero la probabilidad p de seleccionar un movimiento tabú para su reutilización disminuye al acercarse.

Se utilizan dos criterios de probabilidad debido a que las restricciones pueden modelarse con una función de minimización o de maximización. En el primer caso, al modelar un problema con opción de minimización generalmente el valor que se espera obtener en la función objetivo a optimizar es 0 (esto es, cero restricciones no cumplidas) y por ello se debe verificar que el valor inicial de la función objetivo sea mayor que 0 (ya que en este caso se tendría la mejor solución). Por otro lado, cuando se trata de maximizar se espera alcanzar en la función objetivo un valor mayor que 0:

$$p = \begin{cases} \dfrac{\mathcal{GOAL} - f(s_i)}{\mathcal{GOAL}} & \text{si } \mathcal{GOAL} > 0 \\ \dfrac{f(s_i)}{\mathcal{INIT}} & \text{cuando } \mathcal{GOAL} = 0 \text{ y } \mathcal{INIT} > 0, \end{cases} \tag{1}$$

donde: $\mathcal{GOAL}$ es el mejor valor esperado en la función objetivo, $f(s_i)$ equivale al valor de la función objetivo en la i-ésima iteración, y $\mathcal{INIT}$ es el valor inicial de la función objetivo con el cual comienza el algoritmo.

En el Algoritmo 1 se muestra a mayor detalle el funcionamiento del algoritmo BT-CAP. Este asume que una solución s puede ser mejorada haciéndole

Algoritmo 1 Búsqueda Tabú con criterio de aspiración probabilístico.

Inicio

1: **Si** el problema es de maximización **Entonces**
2: $\mathcal{GOAL} \leftarrow$ valor esperado en la función objetivo
3: **Sino**
4: $\mathcal{GOAL} \leftarrow 0$
5: **FinSi**
6: $s \leftarrow$ solución inicial
7: $s^* \leftarrow$ s
8: $\mathcal{INIT}$ = mejor valor en $fo(s)$
9: **Si** $\mathcal{GOAL} = 0$ **y** $\mathcal{INIT} = 0$ **Entonces**
10: ir a **Fin**
11: **FinSi**
12: **Mientras** no se cumpla el criterio de finalización **Hacer**
13: $N(s) \leftarrow$ Generar el conjunto de vecindad
14: $T(s) \leftarrow$ Obtener el conjunto de movimientos tabú
15: $\mathcal{INIT} \leftarrow$ mejor valor en $fo(s)$
16: $m \leftarrow$ el movimiento de $N(s)$ que genera el mejor valor en $fo(s)$
17: **Si** $m \in T(s)$ **Entonces**
18: **Si** $\mathcal{GOAL} > 0$ **Entonces**
19: $p \leftarrow (\mathcal{GOAL} - fo(s)) \div \mathcal{GOAL}$
20: **Sino**
21: $p \leftarrow fo(s) \div \mathcal{INIT}$
22: **FinSi**
23: **Si** aleatorio() $< p$ **Entonces**
24: $s^* \leftarrow s$
25: **FinSi**
26: **FinSi**
27: **FinMientras**

Fin

pequeños cambios, generando soluciones dentro de los vecindad $N(s)$, actualizando la lista tabú $T(s)$, evaluando la función objetivo $f(s)$, con el criterio de aspiración mostrado en las líneas 17 a la 26.

La motivación de crear esta nueva variante del criterio de aspiración surge como una propuesta para variar el proceso de búsqueda al implementar un criterio de aspiración probabilístico que seleccione como válido un movimiento tabú con mayor probabilidad cuando la búsqueda se encuentra lejos de la región factible, de tal manera que se diversifique la búsqueda al iniciar el proceso. Esta probabilidad de aceptar un movimiento tabú disminuye al acercarse a la solución esperada, de tal manera que se intensifique la búsqueda al encontrarse cerca de la región factible.

Por ejemplo, en un problema de minimización el criterio de aspiración emplea la función de probabilidad mostrada en la Ecuación (1.1), siempre y cuando el valor inicial de la función objetivo (señalado como $\mathcal{INIT}$) sea mayor que 0, porque esto significaría que ya se alcanzó la mejor solución posible. El valor de la función objetivo puede corresponder a la suma de restricciones duras o suaves que se desean satisfacer, por ejemplo:

$$p = \begin{cases} \dfrac{fo_i}{\mathcal{INIT}} & \text{cuando } \mathcal{GOAL} = 0 \text{ y } \mathcal{INIT} > 0 \end{cases} \tag{1.1}$$

Si el valor de $\mathcal{INIT}$ es mayor de 0, entonces comienza el proceso de búsqueda y en caso de que un elemento clasificado como tabú mejore la solución actual, entonces se evalúa el posible uso de este elemento de acuerdo con la Ecuación (1.1). Esto significa que al iniciar la búsqueda ($f(s_i) = \mathcal{INIT}$) es más probable utilizar este movimiento clasificado como tabú, en cambio cuando la búsqueda se acerca a la solución óptima ($f(s_i) \approx 0$), es poco probable aplicar el movimiento tabú.

El otro caso consiste en un problema de maximización, en el cual se utiliza el criterio de aspiración con la probabilidad mostrada en la Ecuación (1.2). Aquí es necesario que el usuario defina *a priori* el mejor valor posible a obtener en la función objetivo, que se traduciría en la mejor solución aceptable por el usuario (señalado como $\mathcal{GOAL}$). En este caso es imperativo verificar que el valor de $\mathcal{GOAL}$ sea diferente de 0, porque de lo contrario se traduciría a un problema de minimización:

$$p = \begin{cases} \dfrac{\mathcal{GOAL} - f(s_i)}{\mathcal{GOAL}} & \text{si } \mathcal{GOAL} > 0 \end{cases} \tag{1.2}$$

Una vez realizada la verificación anterior, el criterio de aspiración sigue siendo el mismo que en el caso de minimización, ya que al iniciar la búsqueda ($f(s_i) < \mathcal{GOAL}$) es más probable aplicar un movimiento tabú que mejore la solución actual, pero cuando la búsqueda se acerca a la solución óptima ($f(s_i) \approx \mathcal{GOAL}$) es menos probable que se seleccione el movimiento tabú.

Cabe mencionar que puede ocurrir la situación en la cual todos los vecinos posibles en alguna iteración se encuentren en la lista tabú. En este caso, se aplica el criterio de aspiración por defecto, en el cual el movimiento "menos tabú" o el "más viejo" de la lista es seleccionado.

3 Pruebas y resultados

3.1 Algoritmos desarrollados

Para resolver el problema del ITC se implementaron tres variantes del algoritmo de Búsqueda Tabú:

- **BT-CAD**: Búsqueda Tabú con Criterio de Aspiración por Defecto.
- **BT-CAO**: Búsqueda Tabú con Criterio de Aspiración por Objetivo.
- **BT-CAP**: Búsqueda Tabú con Criterio de Aspiración Probabilístico.

Los tres algoritmos desarrollados consideran la misma sintonización de parámetros:

- Misma solución inicial.
- Misma semilla aleatoria.
- Una única lista tabú con memoria de corto plazo.
- Regla estática para determinar el perido tabú: $t = \sqrt{\textit{número de eventos}}$.
- Búsqueda completa en vecindad $N(s)$.
- Con tres fases:

    1. Creación de una solución inicial.
    2. Resolución de las restricciones duras.
    3. Resolución de la mayor cantidad de restricciones suaves.

Para la creación de la solución inicial se diseñaron diversas estructuras de datos auxiliares que permitiesen evaluar rápidamente las restricciones que violan cada uno de los eventos. Después se diseñó una lista de todos los eventos del problema ordenados de acuerdo con los siguientes criterios:

- Menor cantidad de salones adecuados.
- Mayor cantidad de eventos con participantes en común.
- Mayor cantidad de participantes.

Una de las ventajas de utilizar esta estructura es que con ella se satisfacen automáticamente las restricciones de tipo *ITC-rd1*. Posteriormente, se desarrolló un algoritmo voraz que agenda en primer lugar los eventos más *conflictivos* para resolver así la mayor cantidad de restricciones duras posibles. Las diversas pruebas realizadas con soluciones iniciales aleatorias muestran que las restricciones más difíciles de cumplir son las de tipo *ITC-rd2*, por lo que el Algoritmo 2 genera una solución inicial sin este tipo de restricciones.

Algoritmo 2 Algoritmo voraz para generar una solución parcial inicial.

Inicio

1: Inicializar las celdas de s_i con -1
2: Crear la lista de eventos problemáticos evt_p
3: **Para** cada evento e en evt_p **Hacer**
4: $(i, j) =$ celda vacía con el salón adecuado para el evento e
5: $s_i[i, j] = e$
6: **FinPara**

Fin

Después de múltiples pruebas se llegó a la conclusión que es mejor resolver primero las restricciones duras únicamente, ya que al tratar de resolver las restricciones duras y suaves a la vez, no se logra llegar a una solución factible para algunas de las instancias del problema en el tiempo establecido. Para la etapa de resolución de restricciones duras se modeló la siguiente función objetivo:

$$\min\ z = w \cdot trd2 + trd3 \tag{2}$$

donde: w es igual al máximo número de participantes que cursan un evento multiplicado por 2, $trd2$ equivale al total de restricciones de tipo *ITC-rd2* y $trd3$ es la suma de restricciones de tipo *ITC-rd3*.

El Algoritmo 3 describe con detalle el proceso de resolver las restricciones duras.

Para la etapa de resolución de restricciones suaves se evalúa que cada movimiento plausible no viole restricciones duras y que minimice el número de restricciones suaves, por lo que es la etapa más lenta de todo el proceso. La función objetivo a evaluar durante esta fase es la siguiente:

$$\begin{aligned} \min\ z &= rs1 + rs2 + rs3 \\ \text{sujeta a} &\quad trd = 0 \end{aligned} \tag{3}$$

donde: $rs1$ es igual a la suma del total de restricciones de tipo *ITC-rs1*, $rs2$ es el total de restricciones de tipo *ITC-rs2*, $rs3$ son todas las restricciones de tipo *ITC-rs3* y trd es el total de restricciones duras.

Algoritmo 3 Módulo BT-CAP para resolver todas las restricciones duras.

Inicio

s^* = s_i
Mientras *restricciones duras* $>$ 0 **y** límite de tiempo no alcanzado **Hacer**
  Crear la vecindad $N(s_i)$
  Actualizar la lista tabú $T(s_i)$
  $s_i \leftarrow$ el mejor de $N(s_i)$
  Si $f(s_i) < f(s^*)$ **y** se satisface el criterio de aspiración **Entonces**
    $s^* \leftarrow s_i$
  FinSi
  $i = i + 1$
FinMientras

Fin

Algoritmo 4 BT-CAP para resolver la mayor cantidad de restricciones suaves.

Inicio

s^* = s_i
Mientras límite de tiempo no alcanzado **Hacer**
  Crear la vecindad $N(s_i)$
  Actualizar la lista tabú $T(s_i)$
  $s_i \leftarrow$ el mejor de $N(s_i)$
  Si $f(s_i) < f(s^*)$ **y** se satisface el criterio de aspiración **Entonces**
    $s^* \leftarrow s_i$
  FinSi
  $i = i + 1$
FinMientras

Fin

En el Algoritmo 4 se muestra el proceso de resolver las restricciones suaves.

El hardware utilizado fue una computadora portátil con procesador Intel Centrino a 1.60 GHz, memoria RAM de 2 GB, sistema operativo Ubuntu Linux 10.04 LTS y los tres algoritmos desarrollados sobre la plataforma Java SE 6.0. El tiempo de ejecución fue de 558 segundos (definido por el programa de *benchmark* al ejecutarse en el equipo de prueba) tiempo máximo en el que se debieron ejecutar los algoritmos en busca de una solución factible.

3.2 Resultados

Cabe destacar que los tres algoritmos implementados encuentran una solución factible, esto es, una horario de clases sin restricciones duras violadas.

En la Tabla 1 se muestra el número de restricciones no cumplidas por cada algoritmo para cada una de las 20 instancias del problema. Un valor menor indica mejor calidad de la solución ya que la solución perfecta implica 0 restricciones suaves no cumplidas, y ya que los tres algoritmos resuelven satisfactoriamente todas las restricciones duras, la calidad de la solución obtenida se mide en el número de restricciones suaves resueltas.

Los resultados marcados con (*) en la Tabla 1 indican la mejor solución obtenida para cada una de las 20 instancias del problema. Como se puede observar. El algoritmo BT-CAP supera en todos los casos a las otras dos variantes del algoritmo de Búsqueda Tabú.

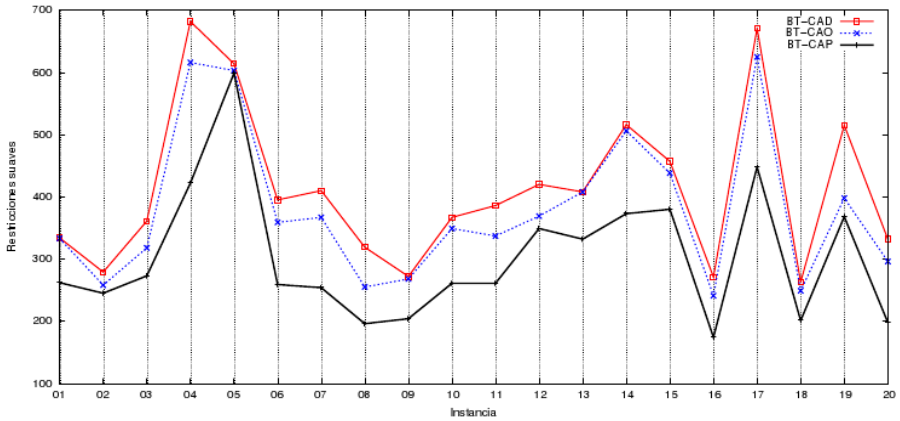

Figura 1: Resultados de las 3 variantes del algoritmo de Búsqueda Tabú en las instancias del ITC.

La Figura 1 despliega gráficamente los resultados mostrados en la Tabla 1, en la cual se puede apreciar con claridad que las soluciones proporcionadas por

Tabla 1: Restricciones suaves no cumplidas por las 3 variantes del algoritmo de Búsqueda Tabú en las instancias del ITC.

	Algoritmo		
Instancia	BT-CAD	BT-CAO	BT-CAP
PROBLEM01	334	333	262*
PROBLEM02	279	258	245*
PROBLEM03	361	318	273*
PROBLEM04	682	616	422*
PROBLEM05	614	603	599*
PROBLEM06	395	459	259*
PROBLEM07	410	367	254*
PROBLEM08	319	255	196*
PROBLEM09	272	268	204*
PROBLEM10	367	349	261*
PROBLEM11	386	337	261*
PROBLEM12	420	369	349*
PROBLEM13	408	408	332*
PROBLEM14	516	506	373*
PROBLEM15	457	438	380*
PROBLEM16	271	241	174*
PROBLEM17	671	625	448*
PROBLEM18	263	249	201*
PROBLEM19	515	398	368*
PROBLEM20	332	296	198*

BT-CAD: Búsqueda Tabú con Criterio de Aspiración por Defecto.

BT-CAO: Búsqueda Tabú con Criterio de Aspiración por Objetivo.

BT-CAP: Búsqueda Tabú con Criterio de Aspiración Probabilístico.

el algoritmo BT-CAP propuesto son mejores, ya que obtiene horarios con menos restricciones no cumplidas.

En la Tabla 2 se muestran el número de restricciones violadas por los nueve participantes que fueron aceptados oficialmente en el concurso ITC [17], así como las restricciones que no se cumplen por el algoritmo de BT-CAP. Los resultados marcados con (*) corresponden a soluciones obtenidas por el algoritmo BT-CAP que superan la mejor solución obtenida por al menos un participante en dicha instancia. Esto significa que en 8 instancias del problema el algoritmo BT-CAP obtuvo una mejor solución que algún participante de la competencia.

Tabla 2: Restricciones no cumplidas por cada participante en las instancias del ITC.

	Participante									
	1	2	3	4	5	**6**	**7**	**8**	**9**	**BT-CAP**
01	45	61	85	63	132	148	178	257	211	262
02	25	39	42	46	92	101	103	112	128	245
03	65	77	84	96	170	162	156	266	213	273
04	115	160	119	166	265	350	399	**441**	408	**422***
05	102	161	77	203	257	412	336	299	312	599
06	13	42	6	92	133	246	246	209	169	259
07	44	52	12	118	177	228	225	99	**281**	**254***
08	29	54	32	66	134	125	**210**	194	**214**	**196***
09	17	50	184	51	139	126	154	175	164	204
10	61	72	90	81	148	147	153	**308**	222	**261***
11	44	53	73	65	135	144	169	**273**	196	**261***
12	107	110	79	119	290	182	219	242	282	349
13	78	109	91	160	251	192	248	**364**	315	**332***
14	52	93	36	197	230	316	267	156	345	373
15	24	62	27	114	140	209	235	95	185	380
16	22	34	**300**	38	114	121	132	171	**185**	**174***
17	86	114	79	212	186	327	313	148	409	448
18	31	38	39	40	87	98	107	117	153	201
19	44	128	86	185	256	325	309	**414**	334	**368***
20	7	26	0	17	94	185	185	113	149	198

3.3 Prueba de Wilcoxon para el problema del ITC

La prueba de Wilcoxon de rangos con signo para un experimento aparejado es un procedimiento estadístico no paramétrico que permite comparar datos con una media teórica cualquiera [26]. Esta prueba consiste en comparar la media de dos muestras relacionadas para determinar si existen diferencias entre ellas.

Para el caso del problema del ITC la hipótesis nula y la hipótesis alternativa corresponden respectivamente a:

- H_0: *No existe diferencia en la calidad de las soluciones obtenidas por el algoritmo BT-CAO y el algoritmo BT-CAP.*
- H_a: *La calidad de las soluciones obtenidas por el algoritmo BT-CAP son mejores que las del algoritmo BT-CAO.*

En la Tabla 3 se despliegan los resultados obtenidos por ambos algoritmos para las 20 instancias del ITC, así como la diferencia entre ambas soluciones para cada instancia. En la última columna se muestra el rango (*rank* o grado) de las diferencias.

Tabla 3: Resultados obtenidos al aplicar la prueba de Wilcoxon a algoritmos BT-CAO y BT-CAP.

Instancia	BT-CAO	BT-CAP	Diferencia	Rango
PROBLEM01	333	262	71	11
PROBLEM02	258	245	13	2
PROBLEM03	318	273	45	5
PROBLEM04	616	422	194	19
PROBLEM05	603	599	4	1
PROBLEM06	459	259	200	20
PROBLEM07	367	254	113	16
PROBLEM08	255	196	59	8
PROBLEM09	268	204	64	9
PROBLEM10	349	261	88	14
PROBLEM11	337	261	76	12.5
PROBLEM12	369	349	20	3
PROBLEM13	408	332	76	12.5
PROBLEM14	506	373	133	17
PROBLEM15	438	380	58	7
PROBLEM16	241	174	76	10
PROBLEM17	625	448	177	18
PROBLEM18	249	201	48	6
PROBLEM19	398	368	30	4
PROBLEM20	296	198	98	15

A continuación se procede a separar el rango de las diferencias en dos grupos según su signo:

- **Diferencias** $< \mathbf{0}$: {–}
- **Diferencias** $> \mathbf{0}$: { 1 2 3 4 5 6 7 8 9 10 11 12.5 12.5 13 14 15 16 17 18 19 20 }

Si H_0 es cierta, se espera que la suma de los rangos de las diferencias menores a 0 sea similar al de las diferencias mayores a 0, y que no se cumpla la condición $T \leq T_0$:

Suma de los rangos de las diferencias menores que 0, T^-: 0.

Suma de los rangos de las diferencias mayores que 0, T^+: 210.

Seleccionando un valor de probabilidad de error $\alpha = 0.01$ con $n = 20$, representando las 20 instancias del ITC, se obtiene un valor para $T_0 = 37$, el cual es el valor crítico para este caso de acuerdo con la tabla 9 mostrada en [26]:

$T_0 = 37$; $T^+ = 210$ y $T^- = 0$; $T =$min(T^+, T^-); en este caso $T = 0$.

La condición para rechazar la hipótesis H_0 es $T \leq T_0$.

Se puede verificar que esta condición es verdadera, por lo que se rechaza la hipótesis nula y se concluye que existe evidencia suficiente de que el algoritmo BT-CAP obtiene soluciones de mejor calidad con el mismo tiempo de ejecución que el algoritmo BT-CAO.

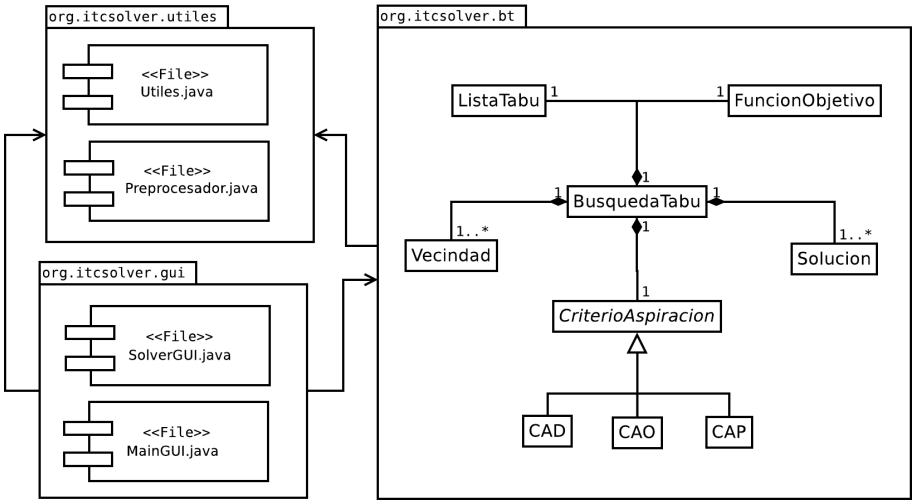

Figura 2: Diagrama de componentes del prototipo desarrollado.

3.4 Prototipo para la solución del problema del ITC

Para la etapa de pruebas se desarrolló un prototipo de software que facilitó este proceso. En la Figura 2 se muestra el diagrama de componentes del paquete `org.itcsolver`, el cual contiene toda la lógica de la aplicación. El prototipo desarrollado consta de tres paquetes: `org.itcsolver.utiles`, `org.itcsolver.gui`, y `org.itcsolver.bt`.

El paquete `org.itcsolver.utiles` contiene el componente `Preprocesador.java`, que se encarga de inicializar las estructuras de datos necesarias para iniciar el proceso de búsqueda. El componente `Utiles.java` contiene métodos auxiliares para el funcionamiento de la aplicación, por ejemplo: conversiones de tipos de datos, operaciones con archivos, funciones de tiempo, entre otras.

El paquete `org.itcsolver.gui` consta de dos componentes: `MainGUI.java` y `SolverGUI.java`, los cuales contienen clases necesarias para mostrar la interfaz gráfica al usuario donde se introducen los datos del problema, y donde se muestra la salida del proceso de búsqueda, respectivamente.

Figura 3: Ventana de configuración de la aplicación.

El paquete `org.itcsolver.bt` contiene las clases que constituyen el motor de búsqueda de la aplicación:

- `BusquedaTabu`: Es la clase principal que utiliza las demás clases del paquete para realizar el proceso de búsqueda. Contiene una instancia de la clase `ListaTabu`, una instancia de la clase `FuncionObjetivo`, una o más instancias de la clase `Vecindad`, una o más instancias de la clase `Solucion`, y una instancia de la clases `CAD`. `CAO` o `CAP`.

- `Solucion`: Contiene la solución actual, opcionalmente puede almacenar la mejor solución encontrada en cada iteración.

- `Vecindad`: Genera una lista de movimientos que pueden aplicarse a la solución en una iteración dada.

- `FuncionObjetivo`: Evalúa el impacto de los movimientos de la vecindad al aplicarlos en la solución actual.

- `ListaTabu`: Almacena los últimos movimientos de la vecindad utilizados para mejorar la solución.

- `CriterioAspiracion`: Clase abstracta que sirve para revocar el estatus tabú de un movimiento que se encuentre en la `ListaTabu`.

- `CAD`: Clase que hereda de `CriterioAspiracion`, constituye el criterio de aspiración por defecto.

- `CAO`: Clase que hereda de `CriterioAspiracion`, constituye el criterio de aspiración por objetivo.

- `CAP`: Clase que hereda de `CriterioAspiracion`, consituye el criterio de aspiración probabilístico propuesto.

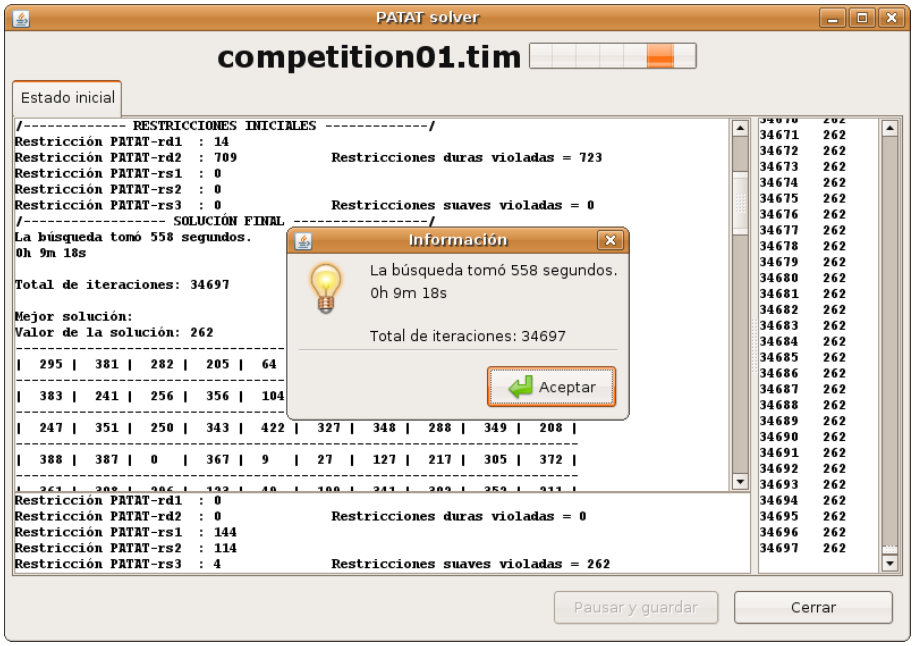

Figura 4: Ventana de despliegue de resultados.

La interfaz de usuario del prototipo se muestra en las Figuras 3 y 4. En la ventana inicial de la aplicación (Figura 3) se solicitan las opciones para configurar los elementos de búsqueda, sintonizar los parámetros del algoritmo, evaluar una solución previa en caso de existir y utilizarla como solución inicial de la nueva búsqueda. Adicionalmente se cuenta con opciones que permiten la depuración de la aplicación y la visualización de estadísticas útiles en casos de prueba.

Los elementos obligatorios a inicializar son los siguientes:

- La instancia del problema que se desea resolver.

- Un valor arbitrario para la semilla inicial. Este valor es necesario debido al uso de probabilidad en el criterio de aspiración y en la creación de vecindades.

- El criterio de finalización. Puede ser por número total de iteraciones, un valor mínimo o máximo en la función objetivo, total de segundos y por un número fijo de iteraciones sin mejora. Para efectos del concurso se utiliza el límite de tiempo fijado en 558 segundos.
- Un tamaño fijo para la lista tabú. Para efectos del concurso se utiliza $t = \sqrt{\textit{número de eventos}}$.
- Si la búsqueda es *agresiva*, esto es, si se va a seleccionar el primer mejor movimiento de la vecindad, o si la búsqueda es *exhaustiva*, indicando que se explorará la vecindad completa.

Una vez configurado el proceso de búsqueda, al presionar el botón `¡Iniciar!` se despliega la ventana mostrada en la Figura 4. Esta ventana muestra el número de restricciones no cumplidas en cada iteración, y dependiendo las opciones de depuración seleccionadas en la ventana de configuración, éstas pueden aparecer en la ventana.

El detalle de las restricciones duras aparece en la etapa de restricciones duras del algoritmo, en la cual las restricciones suaves aparecen en cero ya que no se contemplan aún. Una vez resueltas todas las restricciones duras, para la siguiente etapa se muestra el detalle de restricciones suaves violadas en cada iteración.

El prototipo cuenta con la opción de pausar la búsqueda en cualquier momento, guardando automáticamente la solución encontrada. Al reanudar la búsqueda ésta comienza con la última solución almacenada hasta que se cumpla con el criterio de finalización establecido. Una vez terminado el proceso de búsqueda se despliega una ventana emergente que muestra el tiempo empleado por el algoritmo y el número de iteraciones realizadas.

Esta ventana muestra el número de restricciones no cumplidas en cada iteración, y dependiendo las opciones de depuración seleccionadas en la ventana de configuración puede aparecer:

- El esquema de la solución inicial.
- La sintonización de parámetros.
- Información relativa a la instancia que se está resolviendo.
- Detalle de las restricciones duras y suaves.
- El esquema de la solución obtenida, mostrando cada evento en su respectiva celda en la estructura de solución.
- Una pestaña para cada iteración, mostrando la solución obtenida. No es muy recomendable en el caso de ejecuciones con muchas iteraciones.

4 Conclusiones

El algoritmo de Búsqueda Tabú contiene diversos parámetros que pueden configurarse de múltiples formas, generando soluciones de calidad disímil. En este sentido, se descubre un área de oportunidad en la modificación de un parámetro específico y generalmente ignorado: el criterio de aspiración, de tal manera que permita obtener mejores resultados al ser aplicado al algoritmo original. El Criterio de Aspiración Probabilístico propuesto permite intensificar y diversificar la búsqueda en situaciones clave que mejoran de manera considerable el desempeño global del algoritmo para los problemas presentados.

Es posible utilizar la función de probabilidad propuesta tanto para problemas de maximización (cuando $\mathcal{GOAL} > 0$) como de minimización (cuando $\mathcal{GOAL} = 0$). Para problemas de maximización es necesario conocer *a priori* el mejor valor esperado en la función objetivo, mientras que para problemas de minimización se asume que el mejor valor esperado en la función objetivo es cero. En este caso, se valida que $\mathcal{GOAL} > 0$, ya que si esta expresión es falsa, entonces se habra obtenido la mejor solución esperada.

Con el objetivo de demostrar la robustez del algoritmo de Búsqueda Tabú con Criterio de Aspiración Probabilístico (BT-CAP), se resuelve el problema propuesto en *the First International Timetabling Competition*, en el cual participan los algoritmos más avanzados a nivel internacional. Para realizar las pruebas se desarrolló un prototipo multiplataforma que resuelve satisfactoriamente las 20 instancias del problema ajustándose a las reglas del concurso, obteniendo horarios de clase válidos en todas las instancias.

Se comprueba que el algoritmo de BT-CAP obtiene mejores soluciones que el algoritmo de BT-CAD y el algoritmo BT-CAO, con un 99 % de confiabilidad según la prueba estadística de Wilcoxon, en las 20 instancias del problema.

Es importante mencionar que el algoritmo BT-CAP logra superar en 8 instancias del problema a alguno de los participantes del concurso. Aunque no fue uno de los objetivos de esta investigación, se logró mejorar al menos una solución de cuatro participantes del concurso, con lo que se demuestra la calidad del algoritmo propuesto, obteniendo resultados satisfactorios empleando el algoritmo y a nivel de los mejores especialistas en el área de metaheurísticas y optimización combinatoria.

Como trabajo a futuro se propone el diseño de un experimento factorial para sintonizar los parámetros de mejor manera.

Referencias

[1] Aboytes-Ojeda, M.; Laureano-Cruces, A.; Ramírez-Rodríguez, J. (2013) "Algoritmo de búsqueda tabú para una variante del problema de coloración", *Revista de Matemática: Teoría y Apliciones* **20**(2): 215–230.

[2] Battiti, R.; Tecchiolli, G. (1994) "Simulated annealing and Tabu search in the long run: A comparison on QAP tasks", *Computers & Mathematics with Applications* **28**(6): 1-8.

[3] Cardemil, A. (2002) *Estado del arte y un Algoritmo Tabu Search para el Traveling Tournament Problem*. Tesis de Maestría, Universidad de Buenos Aires, Argentina.

[4] Chávez-Bosquez, O.; De los Santos Torres, G.; Goméz Ramos J.L. (2005) "Búsqueda Tabú aplicada a un problema NP-Completo: Generación de horarios en la DAIS", *Memorias del Congreso Nacional de Informática y Sistemas Computacionales*. Tabasco México.

[5] Chávez-Bosquez, O. (2009) *Búsqueda Tabú Aplicada a un Problema NP-Completo: Timetabling en la DAIS*. Tesis de Maestría, Universidad Juárez Autónoma de Tabasco - División Académica de Informática y Sistemas, México.

[6] Chávez-Bosquez, O.; Pozos-Parra, P.; Lengyel, F. (2011) "Solving the International Timetabling Competition: a deterministic approach", *Fundamenta Informaticae* **113**(1): 1–18.

[7] Chiarandini, M.; Schaerf, A.; Tiozzo, F. (2000) "Solving employee timetabling problems with flexible workload using tabu search", in E.K. Burke & W. Erben (Eds.) *Proceedings of the 3rd International Conference on the Practice and Theory of Automated Timetabling*: 298–302.

[8] Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. (2009) *Introduction to Algorithms*. The MIT Press, Cambridge MA, USA.

[9] Di Gaspero, L. (2002) *Local Search Techniques for Scheduling Problems. Algorithms and Software Tools*. Tesis de Doctorado, Università Degli Stidu Di Udine, Italia.

[10] Frausto-Solis, J.; Alonso-Pecina, F.; Larre, M.; González-Segura, C.; Gómez-Ramos, J. (2006) "Solving the timetabling problem with three heuristics", *WSEAS Transactions on Computers* **5**(11): 2849–2855.

[11] Garey, M.; Johnson, D. (1990) *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W.H. Freeman & Co, San Francisco CA, USA.

[12] Glover, F.; Laguna, M. (1997) *Tabu Search*. Kluwer Academic Publishers, Boston MA, USA.

[13] Glover, F.; Laguna, M. (2013) "Tabu search in analytics and computational science", in: M. Pardalos Panos; D.Z. Du; R.L. Graham (Eds.) *Handbook of Combinatorial Optimization*: 3261–3362.

[14] Gómez Ramos, J. L. (2005) *Algoritmos Genéticos con Diversidad Forzada para la Resolución del Problema de Timetabling Educativo*. Tesis de Maestría, Instituto Tecnológico y de Estudios Superiores de Monterrey, México.

[15] Gutiérrez Andrade, M.A.; De los Cobos Silva, S.G.; Pérez Salvador, B.R. (1997) "Búsqueda tabú: un procedimiento heurístico para solucionar problemas de optimización combinatoria", *Revista En Línea 2*, in: `http://www.azc.uam.mx/publicaciones/enlinea2/1-3.htm`, consultada 4-Jul-2012 12:00 p.m.

[16] Harder, R. (2001) "OpenTS–Java tabu search", in: `http://www.coin-or.org/Ots`, consultada 16-Sep-2012, 4:30 p.m.

[17] ITC (2003) "International Timetabling Competition", in: `http://www.idsia.ch/Files/ttcomp2002`, consultada 7-Sep-2012, 7:50 p.m.

[18] ITC-2007 (2007) "Second international Timetabling Competition", in: `http://www.cs.qub.ac.uk/eventmap`, consultada 16-Sep-2013, 09:40 a.m.

[19] ITC-2011 (2011) "Third international Timetabling Competition", in: `http://www.utwente.nl/ctit/hstt/itc2011`, consultada 16-Sep-2013, 10:00 a.m.

[20] Kendall, G.; Hussin, N.M. (2005) "A tabu search hyper-heuristic approach to the examination timetabling problem at the MARA University of Technology", in: E.K. Burke & M. Trick (Eds.) *Proceedings of the 5th international conference on Practice and Theory of Automated Timetabling*: 270–293.

[21] Larrosa, J.; Meseguer, P. (2003) "Restricciones blandas: modelos y algoritmos", *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial* **7**(20): 69–82.

[22] MNP (2004) "Metaheuristics Network Project", in: `http://www.metaheuristics.org`, consultado el 16/11/2012, 5:20 p.m.

[23] PATAT (2013) "Practice and Theory in Automated Timetabling", in: `http://www.patatconference.org`, consultada 9-Ago-2013, 5:00 p.m.

[24] Schaerf, A. (1999) "A survey of automated timetabling", *Artificial Intelligence Review* **13**(2): 87–127.

[25] Socha, K. (2003) *Metaheuristics for the Timetabling Problem*. DEA Thesis, Université Libre de Bruxelles, Bruselas, Bélgica.

[26] Wackerly D.; Mendenhall W. (2009) *Estadística Matemática con Aplicaciones*. Cengage Learning Editores, México.