Generación Inteligente de Horarios empleando heurísticas GRASP con Búsqueda Tabú para la Pontificia Universidad Católica del Perú

Joseph Gallart Suárez[1], Fernando Alva Manchego[2], Anthony Alama Nole[3], Gissella Bejarano Nicho[4], Manuel Tupia Anticona[5]

Resumen--**El problema de la generación de horarios consiste en la asignación de horas, profesores y aulas para el dictado de las clases, teniendo en consideración diferentes factores como son: disponibilidad de profesores, disponibilidad de aulas, planes de estudio para los diferentes semestres académicos, cruces de horas de dictado entre los diferentes cursos, entre otros. Existe una gran variedad de algoritmos que permiten resolver esta clase de problema. Por las características que presenta la generación de horarios de la Pontificia Universidad Católica del Perú, se decidió emplear algoritmos heurísticos en su solución. El objetivo de este documento es comparar los resultados obtenidos luego de emplear la heurística GRASP y la mejora del mismo con la Búsqueda Tabú. Para la prueba de los mismos, se diseñó el sistema SchedulePowerSoft, que provee una interfaz para el ingreso de los datos requeridos por los algoritmos y la visualización de los resultados generados**

Palabras clave--**Metaheurística, GRASP, Búsqueda Tabú, Problema de asignación de cursos.**

I. INTRODUCCIÓN

En una institución educativa como lo es la Pontificia Universidad Católica del Perú existirá siempre el problema de la asignación de horarios de clase.

Esta labor resulta ardua si se hace manualmente porque requiere prestar mucha atención a la gran cantidad de variables que se toma en cuenta y optimizar su distribución. Estas variables son: la cantidad de aulas disponibles, la capacidad que tiene cada una de ellas, cursos de un mismo ciclo, entre otros.

Este artículo contribuye de manera notoria en brindar una solución al problema de generación de horarios que actualmente se realiza manualmente en la Pontificia Universidad Católica del Perú

Este tipo de problema es conocido en la optimización combinatoria como un Problema de Asignación de Clases (Course Timetabling Problem). Debido a la cantidad exponencial de posibles soluciones, tener una solución exacta es computacionalmente improbable, por ello se justifica usar como solución algoritmos metaheuristicos cuyo objetivo es encontrar una solución suficientemente buena para este tipo de problema.

A continuación se definen los términos que se utilizan a lo largo de este documento, de tal manera que se logre un completo entendimiento del problema.

- *Ciclo*: conjunto de meses donde se dictan los cursos. En el caso de la PUCP[6] por lo general el ciclo dura 4 meses y medio aproximadamente (el ciclo de verano dura 1 mes y medio)
- *Curso*: es una asignatura que requiere de 2 a 4 horas semanales de dictado.
- *Horario*: grupo de cursos que se dictan en un determinado ciclo que no poseen problemas de cruces entre sí.
- *Sesiones de clase*: conjunto continuo de horas en las que se dictan las clases. Un curso puede tener 1 o 2 sesiones a la semana.
- *Aula*: lugar en donde se dictan los cursos.
- *Pabellón:* es una estructura física que contiene un conjunto de aulas.
- *Turno*: rango de horas en cierto momento del día. Existen 2 turnos: mañana (8- 2 pm) y tarde (2- 10 pm).
- *Hueco*: espacio de una hora entre dos clases consecutivas programadas en un mismo salón en un mismo día en el cual no se haya programado ninguna clase.

A continuación se presentan las secciones del artículo. En la sección II se detalla otros algoritmos que resuelven este tipo de problema. En la sección III se explica los algoritmos utilizados. En la sección IV se detalla el diseño de la solución empleada indicando las restricciones principales, secundarias y el modelo de costos. En la sección V se indica la representación de datos usada. En la sección VI se establece los componentes del

[1] Av. Universitaria 1801, San Miguel, Lima 32, Perú 1. E-mail: jgallart@pucp.edu.pe
[2] Av. Universitaria 1801, San Miguel, Lima 32, Perú 1. E-mail: f.alva@pucp.edu.pe
[3] Av. Universitaria 1801, San Miguel, Lima 32, Perú 1. E-mail: a.alama@pucp.edu.pe
[4] Av. Universitaria 1801, San Miguel, Lima 32, Perú 1. E-mail: gissella.bejarano@pucp.edu.pe
[5] Av. Universitaria 1801, San Miguel, Lima 32, Perú 1. E-mail: tupia.mf@pucp.edu.pe (co-autor)

[6] Pontificia Universidad Católica del Perú

algoritmo. En la sección VII se realiza la experimentación numérica dando a conocer los resultados obtenidos. Por último en la sección VIII se brindan las conclusiones llegadas.

II. TRABAJOS PREVIOS

Existe una gran variedad de algoritmos que resuelve este tipo de problemas, como es el caso de los algoritmos genéticos [6], que obtienen una solución aceptable. Otro algoritmo muy usado es el heurístico recocido simulado [7], o también los algoritmos basados en la colonia de hormigas [8], que obtienen también soluciones aceptables.

Por las características particulares del problema que se desea resolver se optó por utilizar el algoritmo de Búsqueda Tabú (Tabu Search), el cual es ampliamente usado en este tipo de problemas (TimeTabling Problem) como se puede ver en [2] y [5]. Sin embargo, este algoritmo funciona en realidad como un optimizador, ya que requiere de una solución inicial como entrada. Mientras mejor sea la solución inicial, mejor serán los resultados que proporcionará la Búsqueda Tabú. Es por ello que se decidió emplear un algoritmo GRASP [1] para poder obtener la solución inicial necesaria. Esta combinación GRASP-Tabú ya ha sido utilizada con éxito en problemas similares (School TimeTabling Problem como en [4] y [9]).

III. ALGORITMOS EMPLEADOS

A. GRASP

GRASP son las siglas para Greedy Randomized Adaptive Search Procedures. Es un algoritmo metaheurístico introducido por Feo y Resende que consiste básicamente de dos fases: construcción y mejora [1].

La fase de construcción se encarga de encontrar una solución factible a través de un procedimiento voraz. Este procedimiento consiste en encontrar, en cada iteración, un elemento del universo y agregarlo a la solución de acuerdo al valor de la función de mérito. Sin embargo, el algoritmo no selecciona el mejor elemento (el que posea el mejor valor de la función de mérito) de todos los posibles sino que emplea una lista restringida de candidatos (LRC). Esto se realiza con el objetivo de reducir el efecto de la miopía en los algoritmos voraces. Para poder formar la LRC se debe tener en consideración tres valores: α (constante de relajación, cuyo valor varía entre 0 y 1), β (mejor valor de la función de mérito) y τ (peor valor de la función de mérito). Luego, la LRC se forma con aquellos elementos que cumplan con la siguiente condición:

$$LRC = \{S_i \in E : \beta \le c(S_i) \le \beta + \alpha(\tau - \beta)\}$$

Fig. 1. Lista Restringida de Candidatos.

Donde:
- E: conjunto de los elementos.
- Si: un elemento del conjunto.
- c: función de mérito.

La función de mérito c debe ser tal que permita determinar cuánto beneficia determinado elemento del conjunto a la solución que actualmente se posee. Dependiendo del tipo de problema que se desee resolver, se plantea qué característica o características de los elementos y de la solución se desea fortalecer (maximizar) o debilitar (minimizar). Teniendo en cuenta esos criterios, se plantea una función de mérito adecuada.

Si se observa detenidamente la fórmula presentada en la Figura 1, podemos ver que la función de la constante α es limitar la cantidad de elementos que formarán parte de la LRC. Si se posee un valor muy bajo de α (muy cercano a 0), la lista se volvería muy pequeña, limitando su cantidad de elementos casi a 1. De contar con un único elemento, este poseería a β como valor de función de mérito; lo que causaría que el algoritmo se volviese miope (característica que se desea limitar). Por otro lado, si el valor de α es muy grande (muy cercano a 1), se tendrá en la LRC a casi todos los elementos del conjunto E, lo cual haría que el algoritmo obtenga soluciones completamente aleatorias. Debido a esta característica, encontrar el valor más apropiado para la constante α resulta de vital importancia.

La segunda fase de GRASP consiste en mejorar la solución brindada por la primera parte del algoritmo. Para ello se emplean algoritmos de búsqueda local como es el caso de la Búsqueda Tabú, la cual se explicará en la siguiente sección.

B. Búsqueda TABÚ

Como se menciona en [3], la Búsqueda Tabú (Tabu Search - TS) es un procedimiento metaheurístico cuya característica distintiva es el uso de memoria adaptativa y de estrategias especiales de resolución de problemas. Como todo algoritmo de búsqueda, TS se basa en la repetición de un determinado proceso hasta que se cumpla una condición de parada.

Para cada iteración del algoritmo, se posee una solución inicial que es la que se desea optimizar. Para esta solución, se genera el vecindario correspondiente, conformado por todas las posibles soluciones resultantes de realizar todos los posibles movimientos predeterminados. Un movimiento viene a ser un cambio en la solución actual que permite obtener una solución diferente a la que se posee. En el contexto del problema que se desea resolver, un movimiento puede ser un intercambio o una inserción de algún elemento del problema en una posición diferente. Más adelante se

proporcionará mayor detalle en relación a los movimientos definidos para el diseño del algoritmo. De este vecindario resultante, se escoge la mejor solución (óptimo local) dependiendo del valor de una función de mérito que determine cuán buena es cada una de las soluciones miembro del vecindario. Este "vecino" seleccionado servirá como solución inicial para la próxima iteración.

Sin embargo, la búsqueda tabú emplea estrategias adicionales para poder obtener una mejor solución en cada iteración. Estas estrategias adicionales consisten en el empleo de una memoria a corto plazo y una memoria a largo plazo. Cada una de estas persigue un objetivo en particular que permite lograr que el resultado brindado finalmente por el algoritmo sea el mejor posible.

La memoria a corto plazo se implementa a través de la denominada lista tabú. En esta lista se almacenan los mejores movimientos que fueron realizados en las diferentes iteraciones, con el objetivo de no volver a repetirlos y caer en un círculo vicioso. Estos movimientos almacenados son denominados elementos tabú. Cabe mencionar que la lista tabú posee un tamaño determinado y, por tanto, los movimientos en ella almacenados no permanecen en ese estado por siempre. Es así que escoger el tamaño adecuado de lista tabú es un factor determinante para la calidad de la solución. Sin embargo, existe otro mecanismo a través del cual un elemento tabú puede ser empleado y son los llamados: criterios de aspiración. Los criterios de aspiración se definen para permitir que un elemento de la lista tabú, que permite obtener una muy buen solución, pueda ser empleado y así lograr una mejor que las que se poseen actualmente como parte del vecindario.

Por otro lado, la memoria a largo plazo está formada por una estructura que permite almacenar la frecuencia en que los distintos movimientos se han ido realizando a lo largo de la ejecución del algoritmo.

Adicionalmente, la búsqueda tabú emplea dos estrategias que, basadas en las memorias de corto y largo plazo anteriormente descritas, permiten obtener una solución final de calidad. Estas estrategias son la intensificación y la diversificación.

La intensificación consiste en explorar a profundidad aquellos vecinos que han brindado una significativa mejoría con respecto a la solución inicial que se poseía en dicha iteración. Esta estrategia emplea la memoria corto plazo para poder lograr su objetivo.

La diversificación, por otro lado, consiste en extender la búsqueda a zonas no visitadas con el objetivo de así encontrar una mejor solución a la que ya se posee. Para ello, esta estrategia emplea la memoria a largo plazo, realizando aquellos movimientos que posean una menor frecuencia de uso.

Finalmente, lo que falta establecer es la condición de parada para la ejecución del algoritmo. En realidad, este factor depende de la forma como se desee diseñarlo, pero entre las condiciones más frecuentes se tienen: número determinado de iteraciones totales, número determinado de iteraciones sin mejora, valor específico de la función de mérito que evalúa la calidad de la solución, entre otros.

IV. DISEÑO DE LA SOLUCIÓN

A. Restricciones Principales

Una solución se considera válida si se cumplen obligatoriamente las siguientes restricciones

- No permitir traslapes entre cursos en una determinada hora en una misma aula.
- Respetar los jueves culturales[7] de la PUCP.
- Para asignar una clase a un salón el número de alumnos de un horario no debe exceder la capacidad del salón.
- Verificar la disponibilidad del profesor.

B. Restricciones Secundarias

Las restricciones secundarias son restricciones que el sistema debería tratar de cumplir, ya que son ellas las que determinan la calidad de la respuesta. Estas restricciones representan las características particulares de la generación de horarios de la PUCP. Son las siguientes:

- Tratar que los cursos que se dictan en los primeros ciclos de facultad (quinto, sexto, séptimo) sean en el turno mañana. (8-2 PM)
- Tratar que los cursos que se dictan en los últimos ciclos de facultad (octavo, noveno, décimo) sea en turno tarde.(2- 10 PM)
- Tratar de no asignar clases a la hora de almuerzo. (12-2 PM)
- Evitar huecos, maximizando el uso de las horas disponibles del salón.

C. Modelo de costos

El modelo de costos a utilizar servirá para darle un peso, un valor de calidad al cumplimiento de las condiciones secundarias, que permita elegir entre realizar un cambio, definidos los movimientos, o mantener una clase en el salón/tiempo en el que se encuentra. Dado que se plantean dos algoritmos, GRASP como generador de la solución inicial y Búsqueda Tabú como optimizador de la misma, se plantean dos modelos de costos, uno para cada algoritmo.

[7] Jueves cultural es un periodo de tres horas (Jueves 12-3PM) durante el cual lo alumnos no reciben dictado de clases.

Finalmente, se emplea un último modelo de costos asociado a la totalidad del horario formado para conocer la bondad del nuevo horario generado en relación a las anteriores soluciones generadas.

El objetivo en el algoritmo GRASP para este caso es el de minimizar f(x), función que representa el desperdicio. Está dada por la Figura 2:

$$f(x) = CapacidadSalon - CantidadAlumnos$$

Fig. 2. Función de mérito GRASP.

Donde:

- CapacidadSalon: máximo número de alumnos que se permite ingresar en el salón sobre el que se ejecuta la función.
- CantidadAlumnos: número de vacantes que el curso tiene asignado.

Este algoritmo verifica previamente que sea un cambio válido (cumpliendo las restricciones principales), y luego determina gracias a la función de mérito (Fig. 2) si es un cambio que mejora la solución.

El algoritmo Búsqueda Tabú valida el resto de condiciones secundarias explicadas anteriormente. El modelo se representa en las Figuras 3 y 4:

$$F = (CI1/CS2 + CI2/CS1) - (CI1/CS1 + CI2/CS2)$$

Fig. 3. Función que mide la calidad de un cambio en función a la cantidad de vacantes.

$$G(i) = \frac{((HI(i) - HF(i)) * ciclo(i) * turno(i) * Factor(i))}{(1 + HA(i))}$$

Fig. 4. Función que mide la calidad de un cambio en base a las restricciones secundarias.

$$T = F - (G(2) - G(1))$$

Fig. 5. Función Tabú Empleada

Donde:

- i: es el índice de clase ha cambiar.
- CI(i): Cantidad de inscritos de la clase i
- CS(i): Capacidad de salón del salón i
- HA(i): Cantidad de horas de cruce de almuerzo de la clase i.
- Factor: Factor que aumenta el peso de la función en caso el curso este en su turno respectivo.
- HI(i): Hora de inicio de la clase i.
- HF(i): hora fin de la clase i.
- Ciclo(i): ciclo al que pertenece la clase i.
- Turno(i): Turno al que pertenece la clase i

La función F busca conocer la calidad de un cambio en relación a la cantidad de alumnos en los salones y la cantidad de vacantes asignadas a las clases que se pretende cambiar.

La función G mide la calidad de la asignación de la clase en el salón/tiempo en que se desea programar, dependiendo del peso de las condiciones secundarias.

La función T, función de mérito del algoritmo Búsqueda Tabú, es la función que se busca maximizar. Se realiza una resta que indica el efecto del cambio con respecto a no realizarlo entre dos clases. De obtenerse T > 0 significa que se obtuvo una mejoría y se procede al cambio; en caso contrario se mantienen las clases en las mismas posiciones.

Los factores (constantes) que se utilizan para aumentar el peso de cada solución fueron calibrados utilizando experimentación numérica.

Los modelos mencionados anteriormente permiten definir si un movimiento entre clases debe llevarse a cabo. El último modelo a representar indica la calidad del horario. Para esto utilizamos un esquema de puntajes. Se recorre la estructura salón por salón, día por día; en cada salón se buscan las clases asignadas y se otorgan los siguientes puntajes.

- Si la clase está en su turno se le asigna 10 ptos.
- Si la asignación respeta la hora de almuerzo se le asigna 5 ptos.
- Si la programación de clases en un salón contiene “huecos” entre clases se le restara 5 pts.

V. REPRESENTACIÓN DE DATOS

Para la representación de datos se usó un arreglo de cuatro dimensiones como se muestra en la figura:

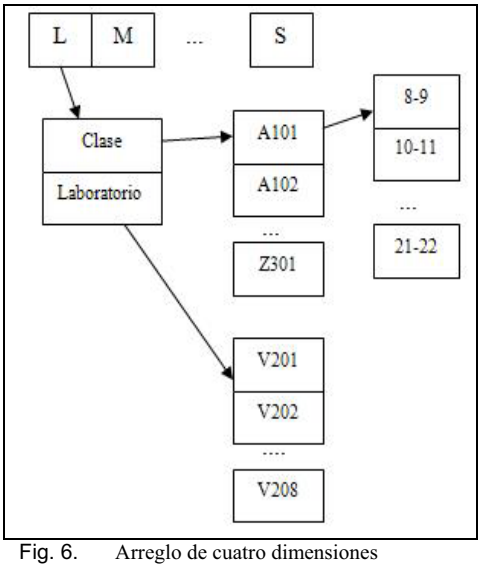

Fig. 6. Arreglo de cuatro dimensiones

La primera dimensión del arreglo representa los días de la semana, la segunda dimensión representa el tipo de aula, la tercera dimensión representa las

aulas de los pabellones, y por último la cuarta dimensión representa las horas de dictado.

VI. COMPONENTES DEL ALGORITMO

A. Construcción de la solución inicial

Para la solución inicial se uso el algoritmo metaheurístico GRASP.

La lista restringida de candidatos (LRC) está dada por los salones candidatos que cumplan la función de mérito planteada en la sección IV-C y por las restricciones principales descrita en la sección IV-A, de esta manera con los salones de la lista restringida de candidatos se construye una solución inicial al problema tratando de minimizar el desperdicio del aula.

B. Mejoramiento de la solución

Luego de obtener la solución inicial se procede a mejorarla utilizando el algoritmo de Búsqueda Tabú.

B.1 Unidades de movimiento

Las unidades de movimiento a emplear son bloques de horas contiguas. Para poder intercambiarlos, se tendrán las siguientes consideraciones:

- Bloque de 2 horas ocupado de una sola clase con bloque de 2 horas ocupado.de otra clase.
- Bloque de 2 horas ocupado con bloque de 2 horas libre.
- Bloque de 3 horas ocupado de una sola clase con bloque de 3 horas ocupado de otra clase.
- Bloque de 3 horas ocupado de una sola clase con bloque de 3 horas libres.

Para cada movimiento que se ejecute con los bloques se debe verificar siempre que se cumplan las restricciones primarias establecidas.

B.2 Lista Tabú

La lista tabú está conformada por los intercambios entre aulas hechos en la ejecución del algoritmo, teniendo la lista un tamaño ingresado como parámetro.

B.3 Memoria

Para la implementación del algoritmo utilizamos la memoria a corto plazo, en ella utilizamos la lista tabú.

B.4 Intensificación y Diversificación

Para poder realizar los intercambios entre los bloques de horas, se han definido casos que permitirán realizar la intensificación y la diversificación. En la tabla I se pueden observar los 7 casos considerados.

TABLA I
CASOS PARA INTERCAMBIOS EMPLEADOS EN EL ALGORITMO TABÚ

Caso 1: • Mismo día • Misma hora • Diferente aula	Caso 2: • Mismo día • Diferente hora • Misma aula
Caso 3: • Mismo día • Diferente hora • Diferente aula	Caso 4: • Diferente día • Misma hora • Misma aula
Caso 5: • Diferente día • Misma hora • Diferente aula	Caso 6: • Diferente día • Diferente hora • Misma aula
Caso 7: • Diferente día • Diferente hora • Diferente aula	

Por cada caso establecido para los cambios se realiza la búsqueda una cierta cantidad de iteraciones dada como parámetro.

B.5 Condición de parada

El algoritmo terminará luego de haber ejecutado los movimientos de intercambio un determinado número de veces que es ingresado como parámetro.

VII. EXPERIMENTACIÓN NUMÉRICA

En este caso se compara el algoritmo GRASP con el algoritmo GRASP-TABU. Solo se aceptan soluciones válidas (que cumplan obligatoriamente con las restricciones principales)

A. Determinación de hipótesis

El criterio que evaluamos es el esquema de puntajes descrito en la sección 6.1

La hipótesis planteada es la siguiente:

$$H_0: \bar{e}_{grasp\text{-}tabu} > \bar{e}_{grasp}$$
$$H_A: \bar{e}_{grasp} \leq \bar{e}_{grasp\text{-}tabu}$$

Fig. 7. Hipótesis planteada

Donde:

- H_0: hipótesis nula, plantea que el algoritmo GRASP-Tabú da mejores resultados que el algoritmo GRASP.
- H_A: hipótesis alternativa, plantea que el algoritmo GRASP da mejores resultados que el algoritmo GRASP-Tabú.
- $\bar{e}_x$: media de los puntajes obtenidos por los algoritmos.

Se espera comprobar que la hipótesis nula es válida.

Nuestra fuente para la realización de este análisis la obtenemos a partir de la ejecución del sistema "SchedulePowerSoft" que fue desarrollado en JAVA (JDK 1.6) y el conjunto de datos reales de los cursos de la especialidad de Ingeniería Informática de la PUCP.

B. Ejecución de pruebas y resultados

Las pruebas se ejecutaron en una computadora Intel Core 2 Duo 2.0 Hz, 2 GB RAM y con sistema operativo Windows XP SP2.

Para la ejecución de las pruebas se tomaron los siguientes parámetros:

- Número de salones (10 y 30).
- Número de profesores (10, 15, 20, 25).
- Número de cursos (20, 25, 30, 35, 40, 45, 50).
- Tamaño de la lista Tabú (5, 15 y 25).

Los números que se indican al lado de los parámetros corresponden a las cantidades empleadas de los mismos en las pruebas.

En la tabla 2 se muestran los resultados obtenidos por el algoritmo GRASP y por el algoritmo GRASP - Tabú. La información empleada para las pruebas fue proporcionada por personal docente encargado de elaborar los horarios en semestres anteriores.

Se generaron 12 combinaciones de los parámetros y por cada una de estas combinaciones se probó cada algoritmo 3 veces para la respectiva comparación del tamaño de la lista Tabú. Cada combinación está compuesta por valores de datos para número de salones, número de profesores y número de cursos.

Los resultados obtenidos comprueban la hipótesis planteada. La media de los puntajes obtenidos por el GRASP-Tabú es mayor a la media de los puntajes obtenidos por el GRASP. Esto se puede explicar por la naturaleza metaheurística que utiliza GRASP-Tabú frente a la naturaleza voraz del GRASP.

TABLA II

PUNTAJES OBTENIDOS POR LOS ALGORITMOS GRASP Y GRASP –TABÚ

# Salones	# Profesores	# Cursos	Tamaño de lista Tabú	Puntaje Algoritmos	
				GRASP	GRASP-TABU
10	10	20	5	140	145
10	10	20	15	150	160
10	10	20	25	140	160
10	10	25	5	155	175
10	10	25	15	165	175
10	10	25	25	200	210
10	10	30	5	205	230
10	10	30	15	200	225
10	10	30	25	180	190
10	15	30	5	160	190
10	15	30	15	195	230
10	15	30	25	200	235
10	15	35	5	205	225
10	15	35	15	190	205
10	15	35	25	155	165
10	15	40	5	210	225
10	15	40	15	275	280
10	15	40	25	220	250
30	20	30	5	300	305
30	20	30	15	295	320
30	20	30	25	310	330
30	20	35	5	300	305
30	20	35	15	260	300
30	20	35	25	305	315
30	20	40	5	350	355
30	20	40	15	400	415
30	20	40	25	350	365
30	25	40	5	340	375
30	25	40	15	335	370
30	25	40	25	385	420
30	25	45	5	380	410
30	25	45	15	355	435
30	25	45	25	410	435
30	25	50	5	360	420
30	25	50	15	390	440
30	25	50	25	420	475

$$\bar{e}_{grasp} = 275.0$$

Fig. 8. Media obtenida con el algoritmo GRASP.

$$\bar{e}_{grasp\text{-}tabu} = 306.25$$

Fig. 9. Media obtenida con el algoritmo GRASP-TABÚ

Luego de la experimentación numérica se comprueba la hipótesis planteada, es decir:

$$H_0: \bar{e}_{grasp\text{-}tabu} > \bar{e}_{grasp}$$

Fig. 10. Hipótesis comprobada

VIII. CONCLUSIONES

Según la experimentación numérica se ha comprobado que un algoritmo GRASP con optimización de Búsqueda Tabú obtiene mejores soluciones que un algoritmo GRASP. Esto se debe a que el método aplicado por GRASP se basa solamente en búsquedas de máximos locales, mientras que en el caso de Búsqueda Tabú, el empleo de la diversificación evita que la búsqueda

se restrinja a máximos relativos, tratando siempre de ubicar máximos globales.

Además, se comprueba que los algoritmos metaheurísticos, si bien no proveen una solución exacta, brindan soluciones lo suficientemente buenas para satisfacer las restricciones del problema y necesidades del usuario.

Finalmente, se puede inferir que utilizar un sistema inteligente que genere automáticamente horarios para una universidad permitirá ahorrar recursos (horas hombre) considerablemente.

REFERENCIAS

[1] Mauricio G.C. Resende, José Luis González Velarde. GRASP: Greedy Randomized Adaptive Search Procedures. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No.19 (2003), pp. 61-76.

[2] Marco Gil Tallavó, Amadís Antonio Martínez, Algoritmo basado en Tabu Search para el problema de asignación de horarios clases, Faraute de Ciencias y Tecnología Volumen 1 – Número 1 Julio 2006.

[3] Belén Melián Batista, Fred Glover, Introducción a la Búsqueda Tabú. Procedimientos Metaheurísticos en Economía y Empresa (E. Crespo, R. Martí y J. Pacheco, coordinadores), Monografías de Rect@, vol. 3, 2007

[4] Marcone Jamilson Freitas Souza, Nelson Maculan, Luiz Satoru Ochi, A GRASP-Tabu Search Algorithm to Solve a School Timetabling Problem. MIC 2001 Proceedings, Porto,Portugal, Julio 2001 pp 53-58.

[5] A Hertz, É. D. Taillard, D. de Werra, Tabu Search, in : E. Aarts and J. K. Lenstra, Local search in combinatorial optimization, 1997, J. Wiley & Sons Ltd., 121-136.

[6] Alberto Colorni, Marco Dorigo, Vittorio Maniezzo, A Genetic Algorithm to solve the timetable problem, Technical Report 90-060, Politecnico di Milano

[7] Ruibin Bai, Edmund K. Burke, Graham Kendall, Barry McCullum, A Simulated Annealing Hyper-heuristic for University Course Timetabling Problem, PATAT 2006, Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling, Brno, August 2006, ISBN 80-210-3726-1, pp345-350.

[8] Krzysztof Socha, Michael Sampels, and Max Manfrin, Ant Algorithms for the University Course Timetabling Problem with Regard to the State-of-the-Art Third European Workshop on Evolutionary Computation in Combinatorial Optimization 2003, Abril 2003, pp334 – 345.

[9] Souza, M. J., Maculan, N., and Ochi, L. S. 2004. A GRASP-tabu search algorithm for solving school timetabling problems. In Metaheuristics: Computer Decision-Making, M. G. Resende, J. P. de Sousa, and A. Viana, Eds. Applied Optimization, vol. 86. Kluwer Academic Publishers, Norwell, MA, 659-672.