Inteligencia Artificial

Informe Final: Balanced Academic Curriculum Problem

Juan Pablo Escalona G.

5 de mayo de 2016

Evaluación

Mejoras 1ra Entrega (10 %):	_____
Código Fuente (10 %):	_____
Representación (15 %):	_____
Descripción del algoritmo (20 %):	_____
Experimentos (10 %):	_____
Resultados (10 %):	_____
Conclusiones (20 %):	_____
Bibliografía (5 %):	_____
Nota Final (100):	_____

Resumen

Balanced Academic Curriculum Problem, también llamado BACP, es un problema que busca asignar los ramos de una malla curricular en diferentes periodos de manera balanceada para que los alumnos puedan cursar exitosamente los ramos dependiendo de la carga (créditos) de estos, cumpliendo restricciones asociadas al la cantidad de ramos y carga en cada periódo. El BACP es un problema recurrente en universidades de todo el mundo, pero también puede ser aplicado en otras áreas como la asignación de carga de trabajo para los empleados de una empresa. En este informe se presenta el estado del arte del BACP resumiendo los diferentes enfoques que se han publicado para resolver este problema desde que fue publicado, comparando las técnicas utilizadas y los resultados.

1. Introducción

Hoy en día las entidades universitarias ofrecen distintas carreras a sus alumnos, estas carreras comprenden una malla curricular con ciertos ramos, que llevan asociado una dificultad. La correcta distribución de los ramos en los periodos académicos puede influir en el éxito o fracaso de los alumnos para terminar su carrera satisfactoriamente. Los ramos pueden comprender de ciertos prerequisitos para ser cursados (e.g., matemáticas 2 necesita matemáticas 1). Además los ramos pueden cambiar su nivel de exigencia para cursarlo exitosamente, lo que se le denominará créditos de ahora en adelante.

Un problema natural que surge es como asignar los ramos de la malla curricular de manera balanceada en su carga, i.e., que los créditos de los ramos de un periodo específico no sean desmedidos, pero que a su vez termine la carrera en un tiempo razonable estableciendo una cantidad máxima y mínima de ramos y créditos por periodo. A esto se le conoce como el Balanced Academic Curriculum Problem (BACP).

El BACP puede ser aplicado en otras áreas como es la asignación de carga de trabajo para los empleados de una empresa basada en turnos, e.g., carga y horarios de las enfermeras de un hospital [8]

En este informe se presenta en forma resumida los distintos métodos, técnicas y enfoques que han publicado los investigadores para representar y resolver el BACP comparando el rendimiento y la calidad de las soluciones. El informe esta organizado de la siguiente manera: En la sección 2 se define el problema y se muestran variantes, en la sección 3 se presentan los estudios realizados anteriormente haciendo comparaciones, luego en la sección 4 se establecen distintos modelos matemáticos y por último en la sección 5 se concluye y plantean futuras líneas de investigación.

2. Definición del Problema

2.1. BACP: el problema

El BACP definido originalmente en [5] busca resolver un problema recurrente para las universidades, asignación los ramos de una malla curricular en distintos periodos, balanceando la carga académica (i.e., similitud entre la carga de todos los periodos), pero a la vez cumpliendo con restricciones de precedencia y límites de exigencias y cantidad de ramos por periodo.

Ha sido demostrado que el BACP es un problema NP-complete [2, 7]

2.2. Definición del Modelo

El BACP definido en [1] comprende los siguientes elementos:

Una malla curricular es un conjunto de ramos que al completarlos se obtiene un título universitario. Cada malla curricular tiene definida una cantidad de periodos en que los ramos deben ser completados.

Los ramos pueden tener distintas dificultades que se relaciona al esfuerzo que un alumno debe aplicar para completar satisfactoriamente dicho ramo, es por esto que se le asocia a los ramos un número de créditos que representa el esfuerzo necesario para completar satisfactoriamente el ramo.

Por otra parte algunos ramos pueden tener prerequisitos que definen una precedencia al ser cursados, es decir, si el ramo β tiene como prerequisito al ramo α, entonces β debe estar en un periodo académico posterior al periodo en que se encuentra asignado α.

La carga académica de un periodo es la suma de los créditos de los ramos que están presentes en dicho periodo. Esta carga académica debe cumplir un máximo para prevenir sobrecarga y un mínimo para ser considerado un estudiante de tiempo completo. Así mismo se establece un máximo y un mínimo de ramos por periodo.

El problema consiste en encontrar una asignación de los ramos en los distintos periodos. Dicha asignación debe satisfacer las restricciones de prerequisitos, máximo y mínimo de créditos por periodo, y máximo y mínimo de ramos por periodo. El óptimo busca balancear la carga ya sea minimizando la máxima carga académica de los periodos [1] o minimizando la diferencias de carga entre los periodos [7].

2.3. Problemas similares y variantes

El BACP es muy similar a problemas como el bin-packing, scheduling y balancing problems, e.g., los prerequisitos del BACP lo hacen similar al scheduling problem asignando precedencia en

intervalos de tiempo. Cae en la clase de problemas en que se tiene un conjunto de objetos de diferentes tamaños los cuales deben ser almacenados en un conjunto de contenedores de capacidad finita, los intervalos de tiempo se asemejan al orden en que los objetos son asignados [7].

La principal variante a este problema es la GBACP presentada en [3]. Esta variante agrega nuevas formulaciones que representan mejor el problema real de las universidades, e.g., se da la posibilidad de compartir ramos de mallas curriculares diferentes, incluir la disponibilidad y preferencia de los profesores al momento de asignar los ramos en periodos específicos y le da la posibilidad a los alumnos de elegir ciertos ramos.

3. Estado del Arte

3.1. Origen del problema

El BACP fue propuesto en 1990 por Hnich et al., [5] y publicado en CSPlib[1] Los primeros en abordar el problema fueron Castro y Manzano [1]. Ellos presentan dos métodos para resolver el problema, el primero consiste en un modelo de programación entera que resuelven utilizando `lp_solve`[2]. El segundo en un modelo basado en restricciones utilizando heurísticas de orden de instanciación de variables y asignación de valores utilizando el lenguaje Oz[3]. Al ser los primeros en abordar el problema, hicieron público en CSPlib tres mallas curriculares las cuales utilizaron en su experimentación. bacp8 una instancia de 8 periodos, bacp10 una instancia de 10 periodos y bacp12 una instancia de 12 periodos, las cuales corresponden a las tres carreras de informática que se imparten en la Universidad Técnica Federico Santa María. Estas instancias se convierten en las utilizadas por la mayoría de los investigadores posteriores para tener resultados comparables.

3.2. Primeros acercamientos: ILP y CP

Los primeros acercamientos realizados por Castro y Manzano, [1] se basaron en la comparación de programación entera en contraste con programación de restricciones utilizando heurísticas de orden de instanciación de variables y asignación de valores. Los tiempos de ejecución para los datasets engregados bacp8, bacp10 y bacp10 fueron los siguientes.

Los resultados se observan en la siguiente tabla muestran el tiempo que tomo a cada método en las distintas instancias de CSPlib, comparando el tiempo que tomo en encontrar la solución óptima.

[1] CSPLib es una librería con múltiples problemas de prueba para solvers de satisfacción de restricciones.
[2] Solver para programación líneal entera mixta. http://lpsolve.sourceforge.net/5.5/
[3] Lenguaje de programación Mozart http://mozart.github.io/

	lp_solve	Oz			
		agrupado por ramo		agrupado por periodo	
		ingenuo	inverso	ingenuo	inverso
8 PERIODOS					
Primera solución	1.7 [s]	0.1 [s]	0.1 [s]	0.2 [s]	0.1 [s]
Mejor solución	1479.7 [s]	∞	0.1 [s]	∞	0.1 [s]
10 PERIODOS					
Primera solución	5.9 [s]	0.2 [s]	0.1 [s]	0.2 [s]	0.1 [s]
Mejor solución	∞	3.6 [s]	0.1 [s]	∞	0.1 [s]
12 PERIODOS					
Primera solución	∞	0.4 [s]	0.5 [s]	∞	0.3 [s]
Mejor solución	∞	∞	0.3 [s]	∞	0.3 [s]

Cuadro 1: Sumario comparativo del rendimiento entre lp_solve y Oz [1]

Notaron de inmediato que las heurísticas utilizadas en Oz eran muy superiores a los resultados que lograba `lp_solve`, Se observa que el modelo de programación entera no logró obtener resultados para el problema de 12 periodos. `lp_solve` no pudo encontrar el resultado óptimo, pues solo llego a 24 créditos en 1626.84 segundos.

3.3. Modelos Híbridos

El año 2002 Hnich et al. [4] publican una nueva técnica para resolver el problema. Presentan un nuevo modelo de CP, pero no se quedan ahí, lo que hacen son experimentos híbridos mezclando las técnicas definidas en [1] con nuevos modelos de CP que definen nuevos espacios de búsqueda.

El nuevo modelo (CP_2) modifica la representación de los periodos en el que se le asigna al ramo i-ésimo un periodo específico utilizando un arreglo (1d), a diferencia del modelo presentado en [1] (CP_1) en que se representaba una matriz (2d) binaria. Lo que se logró fue un modelo mas eficiente al eliminar variables binarias necesarias en las restricciones dada la nueva representación de un arreglo unidimensional disminuyendo el espacio de búsqueda.

Las posibles mezclas de algoritmos híbridos incluyen: $ILP + CP_2$ y $CP_1 + CP_2$. La idea detrás de estos algoritmos híbridos era la de lograr reducir el espacio de búsqueda podando soluciones infactibles. Lo ideal es que el algoritmo híbrido utilice lo mejor de cada uno de los algoritmos que lo componen.

Los resultados que obtuvieron fueron los siguientes:

Modelo	8 periodos		10 periodos		12 periodos	
	tiempo	fallas	tiempo	fallas	tiempo	fallas
ILP	3.45	N/A	4.23	N/A	131.30	N/A
CP_1	58.52	499336	-	-	-	-
CP_2	45.10	56766	-	-	-	-
$ILP + CP_2$	0.81	183	8.44	4445	3.05	525
$CP_1 + CP_2$	0.29	651	0.59	1736	1.09	1539

Cuadro 2: Encontrando una solución óptima

El modelo mas rápido en encontrar una solución optima fue el híbrido entre CP_1 y CP_2, este modelo logró reducir el espacio de búsqueda que compensó el incremento del uso de variables y restricciones debido al uso híbrido de ambos algoritmos.

Este trabajo dio pie a seguir investigando en la línea de modelos híbridos. Uno de ellos es el uso de Algoritmos Genéticos y Propagación de restricciones [6]. En esta investigación utilizan un framework genérico de hibridación. El framework desarrollado les permite especificar la proporción de los métodos, ajustando cuanto tiempo se asigna al GA o al CP. De esa forma pudieron realizar diferentes pruebas cambiando el tiempo que GA o CP tiene destinado en el algoritmo híbrido. De todas formas el principal aporte de esta investigación fue el framework utilizado que les permitió controlar la proporción entre los algoritmos que componen al híbrido.

Dentro de sus resultados en la investigación de GA + CP se observa que logra calcular el óptimo para las instancias bacp8, bacp10 y bacp12 en 15.05, 34.84 y 35.20 segundos respectivamente, mucho mas rápido de lo que logra `lp_solve`.

3.4. Función objetivo alternativa

El 2007 aparece un nuevo ángulo para atacar el problema consiste en minimizar la suma de las distancias medias de la carga de cada periodo y la carga promedio total, i.e.,

$$\min C(p) = \sum_{i=1}^{m} |m\,\alpha_i - w|^p$$

donde α_i es la carga del ramo i, y w es la carga total [7] que se obtiene como:

$$w_i = \sum_{i \in \text{ramos}} \alpha_i$$

En esta investigación comparan 4 casos particulares: $C(1), C(2), C(\infty)$ y el método clásico de [1] C^{max}. Junto con esta nueva función objetivo los autores introducen un método híbrido entre branch-and-bound y Tabu Search como método de búsqueda local. Esto permite encontrar soluciones factibles rápidamente pero no garantizan el óptimo. Además sugieren que dada la naturaleza de las instancias (bacp8, bacp10 y bacp12) las restricciones de máximos y mínimos para la carga por periodo y cantidad de ramos por periodo no es necesario considerarlas, pues las soluciones balanceadas de estas instancias siempre respetan dichas restricciones.

Monette et al., aportan con un generador de instancias, el cual utilizan para generar 100 nuevas instancias mas grandes y complejas que las bacp8, bacp10 y bacp12. En sus experimentos concluyen que el método híbrido de branch-and-bound y Tabu search es mas eficiente para el modelo C^{max}, ya que tabu search logra encontrar óptimos locales rápidamente, para que luego branch-and-bound pueda proseguir mejorando soluciones alternativas. Para el caso de minimizar $C(1)$ proponen el método iterativo como mejor opción. Es por esto que plantean como futuras líneas de investigación el uso de portafolio de diferentes métodos de búsqueda específicos para cada instancia del BACP.

Dentro de la experimentación que realizan con las instancias generadas muestran que las instancias con mayor cantidad de requisitos tienden a ser mas fáciles de responder, más aún, los problemas que tienen entre un 0 % y un 25 % de prerequisitos tienden a ser difíciles. La dificultad disminuye considerablemente cuando se pasa el umbral del 25 % [7]. Esto se traduce en cantidad de restricciones, el espacio de búsqueda del problema inicialmente es muy grande y mientras mas prerequisitos, mas restricciones aparecen. Es fácil entonces reducir drásticamente el espacio de búsqueda del problema y escapar del estado de transición[4].

[4]El estado de transición es el rango en que un problema es mas difícil dada la cantidad de restricciones que el problema presenta.

3.5. Búsquedas locales híbridas

El año 2008 Di Gaspero y Schaerf [3] en donde proponen el GBACP, una variantes más compleja del BACP que busca acercar el problema a la realidad, donde hay ciertos aspectos que BACP no considera, algunos de estos son: los estudiantes pueden elegir alternativas de ramos e.g., ramos de especialidad, las mallas curriculares pueden compartir ciertos ramos, e.g., matemáticas y físicas, los profesores pueden dar su disponibilidad y preferencia para dictar un ramo en periodos específicos.

Para resolver esta variante deciden utilizar métodos de búsqueda local (Tabu Search, Simulated Annealing, Steepest Descent, entre otros), los cuales aplican como primera instancia a los problemas de CSPlib (bacp8, bacp10 y bacp12), los resultados que obtienen son en general muy buenos, con una tasa de éxito de 66.23 % promedio para el bacp8. Los métodos Simulated Annealing, Tabu Search y Dynamic Tabu Search obtuvieron una tasa de exito de 100 % para el bacp8 y bacp10, por los que parecen ser los mas apropiados para abordar este problema.

3.6. Hormigas y rastros de feromonas

Uno de los trabajos mas reciente (2013) sobre el BACP utiliza metaheurísticas de Ant Colony Optimization (AOC) en específico el método BWAS (Best-Worst Ant System). Esta investigación muestra como se puede simular el comportamiento de hormigas para lograr satisfacer las restricciones del problema. Para esto se crea un algoritmo modificado de BWAS para actualizar y mutar rastros de feromonas, así como también reiniciar el proceso de búsqueda cuando este se estanca [8]. Fueron capaces de encontrar óptimos para los problemas bacp8, bacp10 y bacp12, luego de 100 iteraciones el algoritmo no logra encontrar mejores soluciones, pues ya logró el óptimo.

Este método resulto ser muy rápido para hallar las soluciones óptimas, tardando 1.25, 1.92 y 6.37 segundos para el bacp8, bacp10 y bacp12 respectivamente. En su estudio también aplican el BACP a el caso real de su propia casa de estudios, la Universidad Católica de Valparaíso y Universidad de Playa Ancha.

3.7. Comparación de métodos

En la siguiente tabla se muestra una análisis comparativo entre el tiempo que tomo a cada método encontrar una solución optima (o lo mas cercano)[5].

[5]Es necesario destacar que hay una diferencia de hasta 12 años entre el primer método y el mas reciente estudiado en este informe, por lo que parte de la mejora en el rendimiento puede deberse a la evolución del poder de computación en el tiempo.

	bacp8	bacp10	bacp12
Castro y Manzano (2001) [1]			
lp_solve	1459.73 [s]	1626.84 [s] (no óptimo)	∞
Hinch et al. (2002) [4]			
ILP	3.45 [s]	4.23 [s]	131.30 [s]
CLP_1	58.52 [s]	∞	∞
CLP_2	45.10 [s]	∞	∞
$ILP + CLP_2$	0.81 [s]	8.44 [s]	3.05 [s]
$CP_1 + CLP_2$	0.29 [s]	0.59 [s]	1.09 [s]
Lambert et al. (2006) [6]			
$GA + CP$	15.05 [s]	34.84 [s]	35.20 [s]
Di Gaspero y Schaerf (2008) [3]			
Simulated Annealing	0.0042 [s]	0.0429 [s]	0.1764 [s]
Tabu Search	0.0023 [s]	0.0046 [s]	0.0459 [s]
Dynamic Tabu Search	0.0026 [s]	0.0060 [s]	0.0843 [s]
Rubio et al. (2013) [8]			
Best Worst Ant System	1.25 [s]	1.25 [s]	6.37 [s]

Cuadro 3: Comparación de resultados entre todos los métodos estudiados

Es claro entonces que los mejores resultados son los logrados por Di Gaspero y Schaerf [3] utilizando métodos de búsqueda local híbridos. Muy de cerca están los resultados de Hinch et al. Pareciera entonces que los métodos híbridos son el método preferido para encontrar soluciones al BACP. En las investigaciones mas recientes [8] utilizan metaheuristicas de best worst and system, pero sus resultados no son los mas óptimos a pesar de la ventaja tecnológica por ser la investigación mas reciente.

4. Modelo Matemático

4.1. Representación en matriz binaria

El primer modelo clásico es el presentado en [1], en su representación utiliza una matriz binaria para establecer x_{ij} si un ramo i pertenece a un periodo j. De esta manera se puede obtener la carga total de el periodo k-ésimo como la suma de la columna k de la matriz de pertenencia.

Parámetros

m: número de ramos

n: número de periodos académicos

α_i**:** número de créditos del ramo i $\quad \forall i = 1..m$

β**:** carga académica mínima permitida por periodo

γ**:** carga académica máxima permitida por periodo

δ**:** cantidad mínima de ramos por periodo

ϵ**:** cantidad máxima de ramos por periodo

Variables

$$x_{ij} = \begin{cases} 1 & \text{si ramo } i \text{ es asignado al periodo } j \\ 0 & \text{en otro caso} \end{cases} \qquad \forall i = 1..m, \forall j = 1..n$$

Restricciones

Carga académica del periodo j: se obtiene sumando los créditos (α_i) de todos los ramos del periodo actual (x_{ij} al ser variable binaria es la que permite sumar o no según la pertenencia de ramos en los periodos)

$$c_j = \sum_{i=1}^{m} \alpha_i x_{ij} \qquad \forall\, j = 1 \ldots n$$

Todos los ramos i **deben ser asignados a algún periodo** j: Se obtiene sumando la variable binaria para el ramo i, la cual indica si el ramo i esta en el periodo j, los ramos correctamente asignados solo pueden pertenecer a 1 periodo, por lo que la suma de x_{ij} para cada ramo i debe ser 1.

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall\, i = 1 \ldots m$$

El ramo b **tiene como prerequisito al** a: El periodo donde esta asignado el ramo a debe ser inferior al del periodo del ramo b, por lo que se suman todas las x_{ar} hasta el periodo anterior al periodo de b y se iguala a 1 forzando a que el ramo a este en algún periodo entre 1 y $j-1$ cuando el ramo b esta en el j.

$$x_{bj} \leq \sum_{r=1}^{j-1} x_{ar} = 1 \qquad \forall\, j = 2 \ldots n$$

La carga académica máxima esta definida como

$$c = \text{Max}\,\{c_1, \ldots, c_n\}$$
$$c_j \leq c \qquad \forall\, j = 1 \ldots n$$

La carga académica de un periodo no debe pasar los límites

$$\beta \leq c_j \leq \gamma \qquad \forall\, j = i \ldots n$$

La cantidad de ramos de un periodo no debe pasar los límites

$$\delta \leq \sum_{i=1}^{m} x_{ij} \leq \epsilon \qquad \forall\, j = 1 \ldots n$$

Función Objetivo

c es el máximo de las cargas de todos los periodos. La función objetivo minimiza dicha carga máxima.

$$\text{Min } c$$

El espacio de búsqueda para este modelo es una matriz, en que cada elemento (i, j) puede tomar dos valores posibles. Si el tamaño de la matriz es de $m \times n$ entonces el espacio de búsqueda es de: $2^{m \times n}$. Para los casos particulares de bacp8, bacp10 y bacp12 se tienen los siguientes espacios de búsqueda:

Instancia	n	m	Espacio de búsqueda
bacp8	8	46	2^{368}
bacp10	10	42	2^{420}
bacp12	12	66	2^{792}

Cuadro 4: Espacio de búsqueda para el modelo clásico

4.2. Representación en un arreglo

Esta representación disminuye considerablemente el espacio de búsqueda, ya que utiliza solo un arreglo unidimensional para la asignación de ramos a los periodos. Si bien reduce el espacio de búsqueda, la restricción de la carga académica del periodo j es mas difícil de calcular [4]. El modelo es el siguiente:

Parámetros

m: número de ramos

n: número de periodos académicos

α_i**:** número de créditos del ramo i $\quad \forall i = 1..m$

β**:** carga académica mínima permitida por periodo

γ**:** carga académica máxima permitida por periodo

δ**:** cantidad mínima de ramos por periodo

ϵ**:** cantidad máxima de ramos por periodo

Variables

x_i**:** periodo del ramo i $\quad \forall\, i = 1 \ldots m, x_i \in \{1 \ldots n\}$

Restricciones

Carga académica del periodo j: La carga del periodo j es la suma de los créditos de todos los ramos que están presentes en dicho periodo, por lo que se suman todos los α_i siempre que x_i esté en el periodo j-ésimo.

$$c_j = \sum_{i=1}^{m} \alpha_i \mid x_i = j \qquad \forall\, j = 1 \ldots n$$

El ramo b **tiene como prerequisito al** a

$$x_a < x_b$$

La carga académica máxima esta definida como

$$c = \text{Max}\ \{c_1, \ldots, c_n\}$$
$$c_j \leq c \qquad \forall\, j = 1 \ldots n$$

La carga académica de un periodo no debe pasar los límites

$$\beta \leq c_j \leq \gamma \qquad \forall\, j = i \ldots n$$

La cantidad de ramos de un periodo no debe pasar los límites

$$\delta \leq \sum_{i=1}^{m} 1 \mid x_i = j \leq \epsilon \qquad \forall\, j = 1 \ldots n$$

Función Objetivo

$$\text{Min } c$$

La única diferencia es la variable que se utiliza, esto modifica el espacio de búsqueda. existirán m variables, cada una podrá tomar n valores, por lo que el espacio de búsqueda es de n^m

Instancia	n	m	Espacio de búsqueda
bacp8	8	46	8^{46}
bacp10	10	42	10^{42}
bacp12	12	66	12^{66}

Cuadro 5: Espacio de búsqueda para el modelo de un arreglo 2d

En todos los casos, el espacio de búsqueda es menor que en el modelo clásico.

4.3. Minimizar distancias

Este modelo cambia la función objetivo, en este caso se minimiza la suma de las distancias medias entre la carga del periodo y la carga promedio total [7]. En este modelo deciden eliminar las restricciones de número de ramos máximo y mínimo por periodo, así como también la carga máxima y mínima por periodo argumentando que la naturaleza de las soluciones balanceadas de los problemas (bacp8, bacp10 y bacp12) siempre cumplen con dichas restricciones.

Parámetros

m: número de ramos

n: número de periodos académicos

α_i**:** número de créditos del ramo i $\qquad \forall i = 1..m$

w**:** carga total $w = \sum_{i=1}^{m} \alpha_i$

Variables

x_i**:** periodo del ramo i $\qquad \forall\, i = 1 \ldots m, x_i \in \{1 \ldots n\}$

Restricciones

Carga académica del periodo j

$$c_j = \sum_{i=1}^{m} \alpha_i \mid x_i = j \qquad \forall\, j = 1 \ldots n$$

El ramo b tiene como prerequisito al a

$$x_a < x_b$$

Función Objetivo

Se busca minimizar la carga dada una función de distancia media, la expresionismo $|m\alpha_i - w|^p$ calcula la distancia de los créditos del ramo actual ($alpha_i$) y los créditos promedios (w). La función de distancia media se minimiza para encontrar la menor distancia posible entre todos los ramos y el promedio de los ramos.

$$\text{Min } c(p) = \sum_{i=1}^{m} |m\,\alpha_i - w|^p \qquad p > 0$$

En este modelo, se busca minimizar una función de distancia media, esta distancia viene dada por la expresión $|m\,\alpha_i - w|^p$ que depende de p y representa la distancia de la carga de el periodo i con la carga promedio. En el modelo no existe una preferencia a algún p [7]. En sus estudios comparan para los casos de $p = 1$, $p = 2$ y $p = \infty$ pero no encuentran una relación directa entre p y los resultados obtenidos.

El espacio de búsqueda es el mismo que en el modelo anterior. En la investigación [7] muestran ambas representaciones de variables pero no hacen mención al espacio de búsqueda.

5. Representación

La representación matemática que se utiliza es la propuesta en la sección 4.2, pues es mas sencilla la implementación utilizando un arreglo dinámico, además se reduce considerablemente el espacio de búsqueda.

Se utiliza una estructura de datos llamada `bacp_instance` que contiene la definición completa de la instancia del problema, contiene el número de cursos y periodos, cargas máximas y mínimas, número máximo y mínimo de cursos por periodos, un arreglo con los créditos de cada ramo, el número de preerquisitos, un arreglo `x[i]` el cual indica el periodo del ramo i-ésimo. Como extra se agrega un arreglo de prerequisitos de ramos; para esto se definió una estructura llamada `prereq` que contiene un entero a como prerequisito de un b.

Representación matemática	Representación en `bacp_instance`
m	`n_courses`
n	`n_periods`
α_i	`credits[]`
β	`min_load`
γ	`max_load`
δ	`min_courses`
ϵ	`max_courses`
x_i	`period[]`

Cuadro 6: Representación del problema

6. Descripción del algoritmo

6.1. Heurística Greedy para una solución inicial

Una solución inicial balanceada es mejor que una solución inicial muy mala. SA utiliza una solución inicial al azar, pero esta puede ser reemplazada por una solución inicial construida de manera inteligente utilizando un algoritmo greedy.

Para este caso se utiliza el siguiente algoritmo

Algorithm 1 Heuristica Greedy para inicializar Simulated Annealing de BACP

```
1: procedure GREEDY–INITIALIZATION
2:     solution = [] Solución inicial
3:     optimal_load = 1/n_periodos ∑_i^{n_cursos} créditos[i]    Carga optima
4:     i = 0 Periodo inicial
5:     for ramo in ramos do
6:         if ramos periodo i > max_ramos OR créditos periodo i + créditos(ramo) > optimal_load then
7:             i = (i + 1) %n_periods
8:         end if
9:         solution[ramo] = i
10:        ArreglarColisionesPrereq()
11:    end for
12:    return solution
13: end procedure
```

La idea general es obtener un óptimo teórico utilizando la formula:

$$\frac{1}{\text{n\_periodos}} \sum_{i}^{\text{n\_cursos}} \text{créditos[i]}$$

es el promedio de créditos en los periodos, se obtiene sumando el total de créditos de la malla curricular y dividiendolo por el número de periodos.

Conociendo el óptimo teórico, se comienza una solución inicial vacía (línea 5). Luego se itera sobre cada ramo intentando asignarlo al periodo i-ésimo (líneas 6-8). El periodo i se mueve al siguiente periodo si el número de ramos del periodo i es mayor al máximo de ramos por periodo permitido (restricción) o si el número de créditos del periodo i mas los créditos del nuevo ramo exceden el óptimo (línea 6), si esto sucede se mueve i al siguiente periodo disponible, como seguridad se modula el resultado al número total de periodos, por si el algoritmo greedy llega a pasarse del número de periodos que existen, para que siga asignando en el periodo 0 (línea 7).

Una vez calculado el periodo i-ésimo se asigna el `ramo` al periodo j (línea 9).

El procedimiento **ArreglarColisionesPrereq()** (línea 10) busca en el periodo actual posibles colisiones por prerequisitos, si el ramo a es prerequisito del ramo b y ambos quedan asignados en el periodo i, entonces se reasigna el ramo b al siguiente periodo.

El algoritmo greedy si bien no entrega el óptimo, si logra asignar los ramos inicialmente cumpliendo por lo menos con las restricciones de número máximo y mínimo de créditos y ramos por periodo por lo que SA es guiado a una zona de soluciones mas balanceadas.

6.2. Algoritmo Simulated Annealing para el BACP

El procedimiento Simulated annealing realiza múltiples búsquedas locales en diferentes espacios del conjunto de soluciones, a medida que el algoritmo progresa su criterio para aceptar soluciones peores disminuye, i.e. se vuelve mas estricto con el paso del tiempo.

Existen dos iteraciones principales, la iteración por temperatura y la iteración por búsqueda local. La iteración por temperatura permite explorar en diferentes lugares del espacio de búsqueda, mientras que la iteración por búsqueda local permite explotar soluciones locales. En un principio SA permite aceptar soluciones peores en la búsqueda local siempre que se cumpla una regla de aceptación a partir de la temperatura actual, a medida que la temperatura disminuye el criterio de aceptación se vuelve mas estricto.

Algorithm 2 Simulated Annealing for BACP

```
1: procedure SIMULATED–ANNEALING
2:     s_current = solución inicial con heurística Greedy
3:     s_best = s_current Mejor solución
4:     t_current = Temperatura inicial
5:     t_min = Temperatura final
6:     while t_current > t_min do
7:         for iter veces do
8:             vecino = generar_vecino(s_current);
9:             old_cost = costo(s_current)
10:            new_cost = costo(vecino)
11:            if old_cost > new_cost OR e^{-|(new_cost - old_cost)/t_current|} > rand() then
12:                s_current = vecino
13:            end if
14:            if cost(s_best) > new_cost then
15:                s_best = vecino
16:            end if
17:        end for
18:        t_current = t_current × α
19:    end while
20:    return s_best
21: end procedure
```

En la línea 6 se comienza la iteración por temperatura, mientras la temperatura actual sea mayor a la temperatura mínima se repite el proceso. La temperatura actual disminuye en cada iteración según una tasa de cambio dada por α en la línea 18.

La iteración de búsqueda local sucede entre las líneas 7 a la 17. Para la búsqueda local se genera una solución vecina cambiando aleatoreamente un curso de periodo (línea 8), si el costo de esta nueva solución es menor al costo de la solución anterior o si aleatoreamente se decide aceptar una solución peor (línea 11), entonces la solución actual pasa a ser la nueva solución vecina (línea 12).

Por último se mantiene registro de cual ha sido la mejor de todas las soluciones encontradas. Si se encuentra una mejor entonces se actualiza (líneas 14-16).

7. Experimentos

Los parámetros del algoritmo son:

alpha (α): Tasa de reducción de la temperatura

iter: Número de iteraciones locales a realizar en cada temperatura.

t-init: Temperatura inicial

t-min: Temperatura final

Por lo que se realizaron 4 experimentos, dejando 3 parámetros constantes y variando el parámetro restante entre un rango de valores razonables. Para ayudar en este proceso repetitivo, se escribió un script en python capaz de ejecutar múltiples veces una instancia variando el parámetro a elección, luego los resultados son graficados mostrando el valor de créditos/costos encontrado y su exactitud en comparación al óptimo, para cada valor del parámetro en estudio.

Todos los experimentos fueron probados usando los clásicos bacp8, bacp10 y bacp12.

7.1. Experimento 1: α

α es la tasa de cambio de temperatura, esta debe ser un número entre 0 y 1 que asegure que la temperatura disminuya. Mientras mas cercano a 1 mas lenta la tasa de enfriamiento, mientras mas cercana a 0 es mas rápida. En este caso se varia α entre 0.85 y 0.99 en cambios de 0.005. Esto genera 29 valores de α para probar. En cada valor de α se realizan 25 repeticiones del problema para tener una muestra mas dispersa.

Los parámetros fueron los siguientes:

alpha (α): variable [0.85-0.99] en pasos de 0.005

iter: 100

t-init: 1.0

t-min: 0.00001

7.2. Experimento 2: Iteraciones locales

El número de iteraciones hace referencia a cuantas búsquedas locales realiza el algoritmo en cada temperatura. Mientras mas repeticiones realiza este proceso, mas permite moverse entre vecinos. Este proceso tiene una dependencia directa de la temperatura actual, la cual indica si debe aceptar o no una solución peor, en pos de encontrar una posible solución mejor. El numero de iteraciones se hizo variar entre 100 y 600 en pasos de 25, esto genera 21 repeticiones del problema, con 25 repeticiones por cada valor de las iteraciones.

Para este experimento se utilizan los siguientes parámetros

alpha (α): 0.90

iter: variable entre 100 y 600 en pasos de 25

t-init: 1.0

t-min: 0.00001

7.3. Experimento 3: Temperatura

En el caso de la temperatura se tienen dos parámetros que se pueden controlar, pero por la formula que se utiliza para determinar si se acepta una peor solución, la temperatura inicial conviene dejarla fija en 1 y simplemente variar la temperatura mínima para controlar las repeticiones. En este caso se prueban los siguientes valores de `t_min`: 0.001, 0.0001, 0.00001, 0.000001 y 0.0000001

alpha (α): 0.90

iter: 250

t-init: 1.0

t-min: variable entre $10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}$

8. Resultados

El script en python arroja un gráfico que muestra créditos/costo (en rojo) vs el parámetro de estudio y exactitud (en verde) vs el parámetro de estudio.

8.1. Experimento 1: α

Los resultados obtenidos tras 2175 ejecuciones, 725 por cada problema bacp8, bacp10 y bacp12 variando unicamente el parámetro α y dejando todo lo demás constante como:

iter: 100

t-init: 1.0

t-min: 0.00001

Se obtuvieron los siguientes resultados

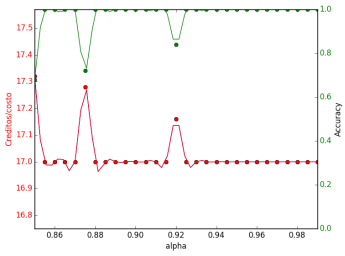

Figura 1: Variación de α en bacp8

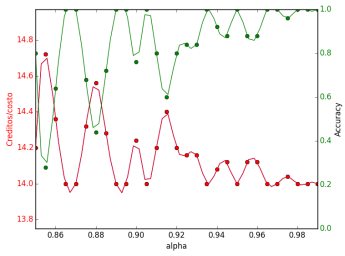

Figura 2: Variación de α en bacp10

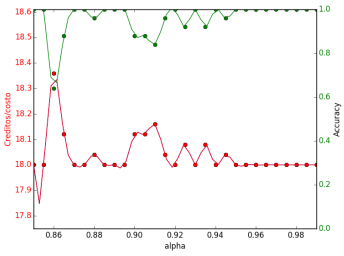

Figura 3: Variación de α en bacp12

Con los siguientes valores promedios:

Instancia	Tiempo	Carga	Accuracy Promedio	Accuracy Final
bacp8	0.3705 [s]	17.0262 [crédito]	97.3793 %	99.8461 %
bacp10	0.3398 [s]	14.1476 [crédito]	85.2414 %	98.9567 %
bacp12	0.7599 [s]	18.0400 [crédito]	96.0000 %	99.7782 %

Cuadro 7: Valores promedios de SA + Greedy sobre bacp8, bacp10 y bacp12 variando α

Observando los tres gráficos se aprecia que mientras mayor es α, mayor cantidad de interaciones de temperatura SA debe realizar, por lo que se nota la tendencia a obtener una exactitud del 100 % cerca de un $\alpha = 0{,}93$ en adelante. Los únicos puntos anormales son el 0.85, 0.875, 0.92 los cuales presentaron un promedio de 17.25 créditos en las 25 repeticiones de la instancia en cada punto.

En este caso del bacp10 se comporto de manera mas errática, pero con una clara tendencia a llegar a los 14 créditos a medida que α aumenta. Nuevamente se observa que los puntos 0.85, 0.875 y 0.92 tienen un peor rendimiento en encontrar el óptimo.

Para el bacp12 el comportamiento fue similar al bacp8, un poco mas violento hasta antes del 0.95 pero la tendencia a estabilizarse es la misma y termina con una exactitud cercana 100 %.

Con respecto a los tiempos promedios, tanto el bacp8 como el bacp10 les tomo solo 0.35 [s] en encontrar soluciones, mientras que el bacp12 fue de cerca del 0.7 [s], aproximadamente el doble de tiempo. Sin embargo la peor exactitud fue la entregada por bacp10, con un 85 % promedio, mientras que bacp8 y bacp12 están ambas por sobre el 95 % de exactitud promedio. De todas formas las exactitudes totales de todas las instancias fueron superiores al 98 %.

8.2. Experimento 2: Iteraciones locales

Los resultados obtenidos tras 825 ejecuciones, 275 por cada problema bacp8, bacp10 y bacp12 variando unicamente el parámetro `iter` y dejando todo lo demás constante como:

α**:** 0.90

t-init: 1.0

t-min: 0.00001

Se obtuvieron los siguientes resultados

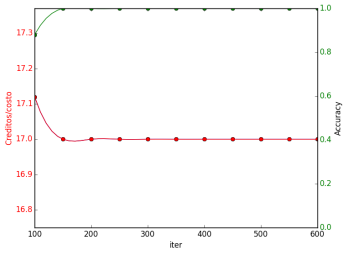

Figura 4: Variación de iteraciones locales en bacp8

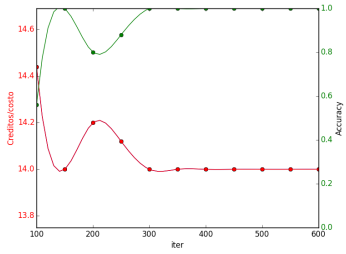

Figura 5: Variación de iteraciones locales en bacp10

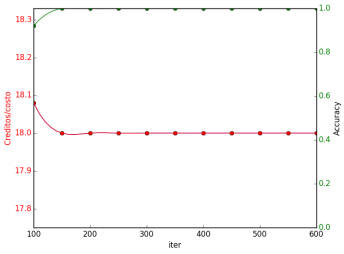

Figura 6: Variación de iteraciones locales en bacp12

Con los siguientes valores promedios:

Instancia	Tiempo	Carga	Accuracy Promedio	Accuracy Final
bacp8	0.6107 [s]	17.0109 [crédito]	98.9090 %	99.9359 %
bacp10	0.5700 [s]	14.0691 [crédito]	93.0910 %	99.5089 %
bacp12	1.25711 [s]	18.0073 [crédito]	99.2730 %	99.9595 %

Cuadro 8: Valores promedios de SA + Greedy sobre bacp8, bacp10 y bacp12 variando `iter`

Lo primero que se observa en este experimento es que los valores se llega rápidamente a soluciones optimas a medida que la cantidad de iteraciones locales aumenta. Incluso en la instancia bacp10, la cual en el experimento anterior resulto ser la mas errática, en este caso solo tiene los valores 100, 200 y 250 incorrectos, ya a partir de las 300 iteraciones logra llegar casi siempre al óptimo.

Por otro lado las instancias bacp8 y bacp12 tienen un comportamiento muy similar, con tan solo 150 iteraciones en adelante se obtienen resultados óptimos. La diferencia entre ambos se produce en el tiempo de ejecución, bacp12 le toma el doble de tiempo que a bacp8. Resalta la exactitud promedio de un 99 % en ambos casos, y de casi el 100 % en la exactitud total.

bacp10 es el único que tuvo menor exactitud promedio de 93.091 %, pero aun así fue mejor que en el experimento 1 donde solo obtuvo un 85.2414 %.

Cabe resaltar que en este experimento se esta utilizando el mejor valor encontrado para α en el experimento 1. Lo suficientemente bueno como para entregar resultados significativos, pero que a su vez no sea demasiado costoso de ejecutar. Se opto por usar el 0.9 para α.

8.3. Experimento 3: Temperatura

Los resultados obtenidos tras 375 ejecuciones, 125 por cada problema bacp8, bacp10 y bacp12 variando unicamente el parámetro `t-min` y dejando todo lo demás constante como:

α**:** 0.90

iter: 250

t-init: 1.0

Se obtuvieron los siguientes resultados

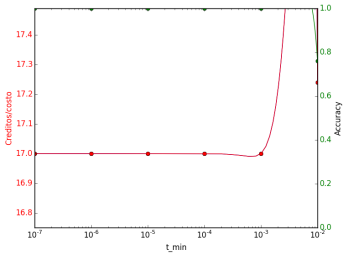

Figura 7: Variación de temperatura mínima en bacp8

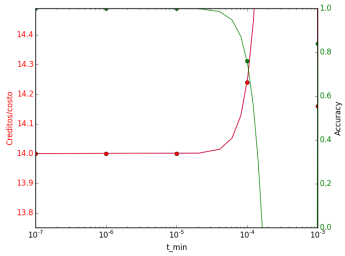

Figura 8: Variación de temperatura mínima en bacp10

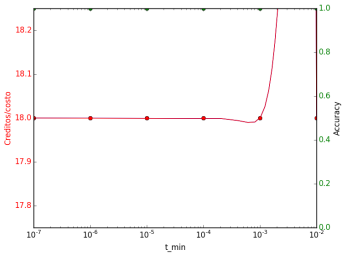

Figura 9: Variación de temperatura mínima en bacp12

Con los siguientes valores promedios:

Instancia	Tiempo	Carga	Accuracy Promedio	Accuracy Final
bacp8	0.3882 [s]	17.0400 [crédito]	96.0000 %	99.7653 %
bacp10	0.4219 [s]	14.0800 [crédito]	92.0000 %	99.5089 %
bacp12	0.7914 [s]	18.0000 [crédito]	100.0000 %	100.0000 %

Cuadro 9: Valores promedios de SA + Greedy sobre bacp8, bacp10 y bacp12 variando `iter`

En este experimento el gráfico se lee al revés, los valores mas pequeños para `t_min` representan una mayor iteración de temperaturas, por lo que los resultados deberían acercarse mejor a una solución óptima. La temperatura mínima esta en escala logarítmica.

Nuevamente bacp8 y bacp12 tienen un comportamiento muy similar, solo fallan en el orden de los 10^{-2}, ya desde los 10^{-3} logran llegar a soluciones optimas siempre. bacp10 solo logra estabilizarse al llegar al orden de los 10^{-5}.

En este experimento se utilizan los mejores parámetros de los dos experimentos anteriores, $\alpha = 0{,}90$ y `iter` $= 250$. Por lo que la calidad del experimento actual debiese ser superior. Los tiempos de ejecución para el bacp8 y bacp10 están cerca de los 0.4 [s] mientras que nuevamente bacp12 llega al doble con 0.4 [s]. Este parece ser un patrón recurrente en todos los experimentos, bacp8 y bacp10 toman tiempos similares mientras que bacp12 toma el doble de tiempo. El otro patrón recurrente es el comportamiento similar entre bacp8 y bacp12, pues suelen poseer para los mismos parámetros resultados similares.

Con respecto a las exactitudes es notable el 100 % obtenido en la instancia bacp12, muy de cerca bacp8 y nuevamente bacp10 resulta ser el que posee menor exactitud con un 92 %.

8.4. Resultados generales

Un patrón interesante en los tres experimentos es que las instancias bacp8 y bacp12 suelen tener un mejor comportamiento, logrando llegar a soluciones de muy buena calidad rápidamente. Por otra parte el bacp10 tiende a ser mas inestable, sus variaciones entre los diferentes parámetros son mucho mayores. En todos los experimentos bacp10 obtiene la menor exactitud promedio. Tanto bacp8 como bacp12 tienen un promedio de 17 y 18 créditos por periodo mientras que bacp10 solo tiene 14, se entiende entonces que bacp10 tiene una mayor cantidad forma de asignar los cursos por lo que el espacio de búsqueda es mas irregular, pequeñas variaciones pueden lograr una gran deferencia en el resultado.

Con respecto a los parámetros en si, α parece ser el mas irregular, pues los cambios en este arrojan resultados muy dispersos. Las iteraciones locales y temperatura tienden a ser mucho mas estables, de todas formas cabe notar que el experimento 2 y 3 fueron ejecutados utilizando los mejores resultados del experimento anterior, esto podría significar una mejor sintonización de los parámetros siguientes.

En general los resultados obtenidos son comparables con los de [3] quienes implementaron simulated annealing con resultados excepcionales. Si bien ambos son inferiores a 1 segundo, en [3] utilizan software especializado en búsquedas locales que ofrecen estructuras especificas para la resolución de este tipo de problemas.

9. Conclusiones

En este informe se presentó el estado del arte sobre el problema BACP. Se resumió la evolución en los métodos que han surgido para resolver este problema. Una de las mejores formas de resolver este problema consiste en utilizar métodos híbridos utilizando heurísticas inteligentes con búsquedas locales (Simulated annealing, Tabu Search y Dynamic Tabusearch) [3] que permiten guiar a los algoritmos a resultados óptimos rápidamente.

Dentro de los posibles modelos, el criterio de minimización del máximo de las cargas de todos los periodos permite resolver el problema, no es el mas óptimo para diferenciar soluciones. El criterio de minimización de distancias medias propuesto en [7] resulta ser mas eficaz al momento de generar distintas soluciones.

Estos nuevos modelos y técnicas híbridas dan la posibilidad a la aparición de variantes mas complejas como lo es el GBACP propuesto en [3, 2].

Como futuras líneas de investigación sería interesante ver el comportamiento de otras técnicas completas, la mayoría utilizan híbridos de locales con CP. Quizás probar con Backtracking o Forward Checking a pesar de los grandes espacios de búsqueda. Por otra parte se demostró que los métodos híbridos son un buen acercamiento para resolver este problema, sería interesante realizar nuevas mezclas híbridas por ejemplo GA con búsquedas locales como Tabu Search o Simulated Annealing.

Con respecto a la implementación de Simulated Annealing con Greedy para este problema se observa que es un algoritmo excelente para resolver el problema. Los resultados obtenidos tienen una exactitud cercana al 99 % y un tiempo de ejecución que por lo general no supera 1 segundo, muy comparables con los presentados en la literatura.

El algoritmo presenta una serie de parámetros que permiten realizar una sintonización interesante. Además la representación elegida es muy sencilla de implementar en forma de arreglo dinámico de heap, lo cual permite tiempos de acceso en un arreglo de orden constante $\mathcal{O}(1)$. El algoritmo greedy fue de gran ayuda para llevar al algoritmo simulated annealing a un sector del espacio de búsqueda que tiende a tener buenos resultados. El algoritmo greedy tiene una orden de $\mathcal{O}(n^2)$, pues asignar los cursos a un periodo es líneal de orden $\mathcal{O}(n)$, pero revisar si existen colisiones implica una revisión extra.

Como futuras líneas de trabajo en esta implementación seria interesante mejorar la eficiencia de varios algoritmos, en especial la inicialización greedy puede ser vastamente mejorada, por

ejemplo el calculo de los créditos no tiene por que ser calculada en cada iteración, podría ser asignada localmente y utilizar incrementación por cada iteración de asignación. Otro aspecto que podría ser mejorado es una tabla hash para los prerequisitos de un curso. Actualmente se utiliza un arreglo dinámico con estructuras que definen cursos a y b, por lo que buscar los prerequisitos de a tiene un orden $\mathcal{O}(n)$, pues se debe recorrer la lista completa de prerequisitos buscando las estructuras en que a sea el curso que estamos buscando. Una tabla hash podría ser indexada por el curso, así acceder al elemento en cuestión seria inmediato. Otra forma de hacerlo seria un arreglo de prerequisitos, en que cada indice es el numero de un curso, y dentro una lista de prerequisitos. Este método seria muy rápido de acceder a la lista de prerequisitos de un curso en especifico.

10. Bibliografía

Indicando toda la información necesaria de acuerdo al tipo de documento revisado. Las referencias deben ser citadas en el documento.

Referencias

[1] Carlos Castro and Sebastian Manzano. Variable and value ordering when solving balanced academic curriculum problems. *CoRR*, cs.PL/0110007, 2001.

[2] Marco Chiarandini, Luca Di Gaspero, Stefano Gualandi, and Andrea Schaerf. The balanced academic curriculum problem revisited. *Journal of Heuristics*, 18(1):119–148, 2012.

[3] Luca Di Gaspero and Andrea Schaerf. Hybrid local search techniques for the generalized balanced academic curriculum problem. In MaríaJ. Blesa, Christian Blum, Carlos Cotta, AntonioJ. Fernández, JoséE. Gallardo, Andrea Roli, and Michael Sampels, editors, *Hybrid Metaheuristics*, volume 5296 of *Lecture Notes in Computer Science*, pages 146–157. Springer Berlin Heidelberg, 2008.

[4] Brahim Hnich and Zeynep Kiziltan. Modelling a balanced academic curriculum problem. In *Proceedings of CP-AI-OR-2002*, pages 121–131, 2002.

[5] Brahim Hnich, Zeynep Kiziltan, and Toby Walsh. CSPLib problem 030: Balanced academic curriculum problem (bacp). `http://www.csplib.org/Problems/prob030`.

[6] T. Lambert, C. Castro, E. Monfroy, and F. Saubion. Solving the balanced academic curriculum problem with an hybridization of genetic algorithm and constraint propagation. In Leszek Rutkowski, Ryszard Tadeusiewicz, LotfiA. Zadeh, and JacekM. Zurada, editors, *Artificial Intelligence and Soft Computing – ICAISC 2006*, volume 4029 of *Lecture Notes in Computer Science*, pages 410–419. Springer Berlin Heidelberg, 2006.

[7] Jean noël Monette, Pierre Schaus, Stéphane Zampelli, Yves Deville, and Pierre Dupont. A cp approach to the balanced academic curriculum problem, 2007.

[8] José-Miguel Rubio, Wenceslao Palma, Nibaldo Rodriguez, Ricardo Soto, Broderick Crawford, Fernando Paredes, and Guillermo Cabrera. Solving the balanced academic curriculum problem using the aco metaheuristic. *Mathematical Problems in Engineering*, 2013:Article ID 793671, 8 p., 2013.