PROPUESTA DEL MÉTODO AGM PARA EVALUAR PROBLEMAS ALGORÍTMICOS MEDIANTE GRAFOS DE APOYO EN LA CONTRUCCIÓN DE EVALUACIONES EN PRIMER AÑO

AUTORES

RODRIGO COFRÉ LOYOLA
rcofre@spock.ucm.cl

ANGÉLICA URRUTIA SEPÚLVEDA
aurrutia@spock.ucm.cl

CARLOS JARA ROJAS
cjara@ganimides.ucm.cl

DEPARTAMENTO DE COMPUTACIÓN E INFORMÁTICA
FACULTAD DE CIENCIAS DE LA INGENIERÍA
UNIVERSIDAD CATÓLICA DEL MAULE
CASILLA 617, TALCA CHILE

MAYO 2000

RESUMEN

Este trabajo está inserto en la temática de la construcción y complejidad ciclomática de algoritmos que debemos dar a los alumnos de primer año de la asignatura de Introducción a la Computación y Lenguajes de Programación, de la carrera de Ingeniería Civil Informática.

Nuestra propuesta del método AGM, se basa en el uso de grafos a partir de un algoritmo dado, para medir su complejidad ciclomática mediante una Matriz de Conexión y usando esta técnica en la elección de algoritmos que se incluirán en una evaluación, sea esta parcial o global de un curso de primer año, y permitiéndonos también determinar las conductas de entrada cuando este sea pre-requisito de otra asignatura.

Presentamos algunos resultados obtenidos en la aplicación de este método en construcción de evaluaciones escritas.

Palabras claves: Evaluación algorítmica, Complejidad Ciclomática, Matriz de conexiones, Grafos de flujo.

INTRODUCCIÓN.

Si bien la Complejidad Ciclomática se usa en Ingeniería de Software, también puede ser utilizada en la generación de instrumentos de evaluación de algoritmos para alumnos de un primer curso de algoritmos.

Una de las tareas más dificultosas para un docente que imparte una asignatura inicial de algoritmos es la construcción de instrumentos de evaluación, mas aún, si son para alumnos de primer año de la especialidad. En la asignatura de Introducción a la Computación y Lenguajes de Programación, se ha aplicado esta técnica para medir la complejidad de los algoritmos que utilizaremos en el logro de los objetivos planteados al inicio de la asignatura.

Nuestra inquietud se ha planteado a partir de los errores que pueden darse desde el primer momento que se eligen algoritmos, los cuales pueden ser parte de un proceso de evaluación para cumplir algún objetivo. Los algoritmos pueden estar planteados de forma errónea e imperfecta, como así también estar sobredimensionados o subdimensionados después de ser usados como instrumentos de evaluación. El desarrollo de instrumentos de evaluación, ha de ir acompañado de una actividad que garantice la calidad. Para solucionar esta problemática proponemos el método AGM.

Esta propuesta la haremos en los siguientes pasos: conceptos básicos, propuesta del método AGM, aplicación del método AGM, conclusiones y bibliografía.

1. CONCEPTOS BÁSICOS

Antes de explicar el método propuesto en este documento, mostraremos las notaciones que se emplean en este trabajo: Representación de Grafos de Flujo, Estructuras Lógicas, Matriz de Conexión, Complejidad Ciclomática.

1.1-. Representación de Grafos de flujo: Para ello, cada estructura algorítmica tiene su respectiva representación gráfica, la cual se muestra en la figura 1. [1][2]

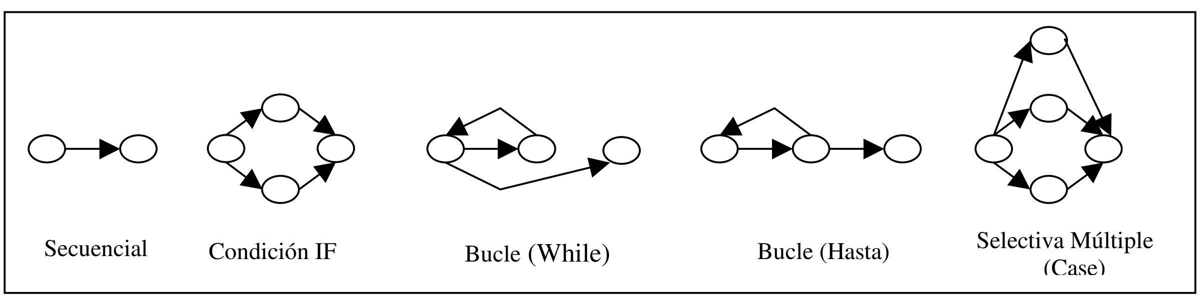


Figura 1. Estructuras en forma de grafo**.**

En estas estructuras, un nodo o círculo representa a una o más sentencias de un procedimiento o función, las cuales no tienen bifurcaciones. Las flechas del grafo de flujo se denominan *enlaces* o *aristas*, éstas representan el flujo de control. Una arista debe terminar en un nodo, aunque el nodo no tenga ninguna sentencia procedimental. Las áreas delimitadas por aristas y nodos de denominan *Regiones*. Cada nodo que contiene una condición se denomina *Nodo Predicado*.

En estas notaciones, el nodo de la izquierda (según la figura Nº1) representa el inicio de una estructura de control, en cambio el nodo de la derecha representa el fin de estas estructuras. De esta manera, los nodos interiores de estas notaciones representan las sentencias que poseen las estructuras de control de un algoritmo, ya sean sentencias simples o compuestas.

1.2-. Estructuras Lógicas:Una sentencia condicional, ya sea en una estructura SI, MIENTRAS o HASTA, se vuelve compuesta cuando intervienen operadores lógicos OR, AND, NOR o NAND. Para ilustrar este tipo de notaciones, a continuación se representan las sentencias: [1][2]

SI (A or B) *Entonces* X *Sino* Y	*SI* (A and B) *Entonces* X *Sino* Y

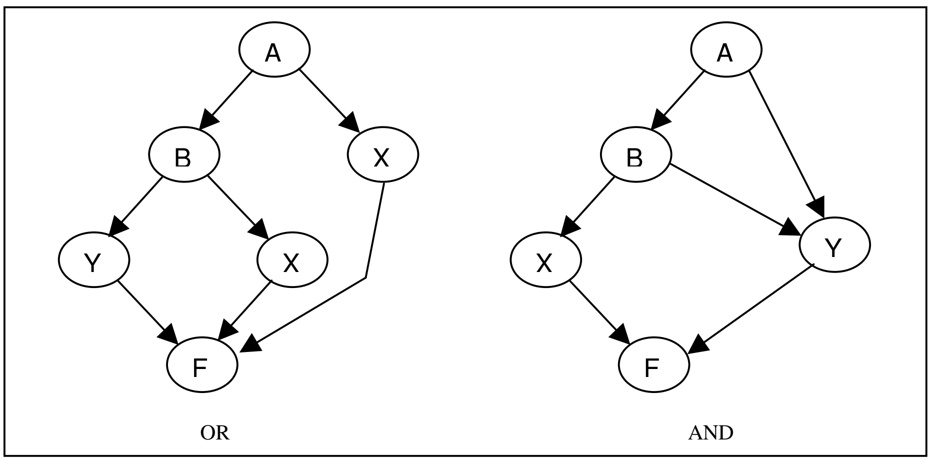
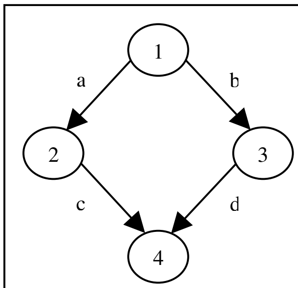


Figura 2. Grafos de condicionales con OR y AND

Donde A y B son operandos de los operadores lógicos OR y AND, X e Y son sentencias a realizar según el valor de la condición y F es el cierre o fin de la sentencia SI (FinSi).

1.3-. Matriz de Conexiones: Una matriz de grafo es una matriz cuadrada de rango igual al número de nodos que posee dicho grafo de flujo, donde cada columna o fila corresponde a un nodo específico. Las valores (entradas) de la matriz son las conexiones o aristas que existen entre los nodos. [1][2]



Matriz de Conexiones

	1	2	3	4
1		a	b	
2				c
3				d
4				

Figura 3. Representación de la matriz de conexiones apartir de un grafo.

1.4-. Complejidad Ciclomática: Para medir cuantitativamente la complejidad lógica de un algoritmo, necesitamos el grafo de flujo y la matriz de conexiones. Aunque esto se puede determinar sin desarrollar el grafo, contando todas las sentencias condicionales del código, se puede calcular la complejidad ciclomática V(G) de tres formas: [1][2]

- V(G) es igual al número de regiones de un grafo.
- $\boxed{V(G) = E - N + 2}$

  donde E es el número de aristas y N es el número de nodos del grafo G.

- $\boxed{V(G) = P + 1}$

donde P es el número de nodos predicado contenidos en el grafo G.

También podemos determinar la complejidad ciclomática de un grafo de flujo a través de su matriz de conexiones. Para ello, se reemplaza cada entrada del grafo por un 1 si existe conexión y por un 0 (cero) si no existe conexión. Para fines de simplicidad, el 0 (cero) se omite. Luego se suman las entradas en cada fila y se le resta 1 (uno), si no se registran entradas, entonces el valor será 0 (cero). Finalmente, se suman estos resultados, y al total se le suma uno para obtener la complejidad ciclomática. Esto se muestra en la figura 4.

	1	2	3	4
1		a	b	
2				c
3				d
4				

	1	2	3	4
1		1	1	
2				1
3				1
4				

Conexiones

1: 2 – 1 = 1
2: 1 – 1 = 0
3: 1 – 1 = 0
4: 0
+____________

Complejidad Ciclomática V(G) = 1 + 1 =2

Figura 4. Complejidad ciclomática a partir de la matriz de conexión.

2.- PROPUESTA DEL MÉTODO AGM.

El método propuesto, el cual llamaremos AGM, se compone de tres etapas: Algoritmos, Grafos y Matriz. Con la aplicación de cada una de ella se logra la evaluación de la complejidad que presenta un determinado planteamiento algorítmico, la figura 5 muestra el método.

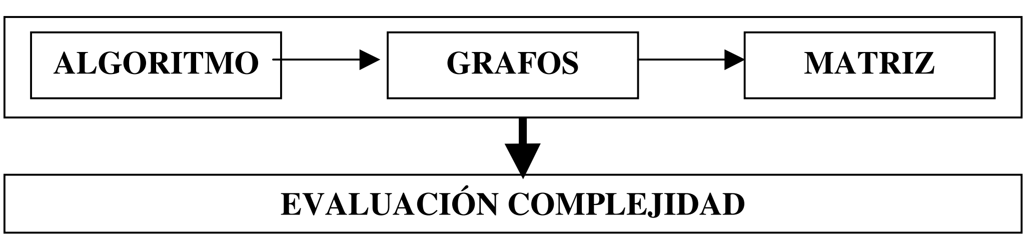


Figura 5. propuesta de método AGM.

Cada una de estas etapas la definimos de la siguiente forma:

2.1-. Algoritmos: Esta primera etapa está orientada a la recopilación de problemas algorítmicos que serán empleados en la evaluación académica. Estos deben cubrir un gran diversidad de problemas abarcando las temáticas del curso. De esta forma garantizar satisfactoriamente el cumplimiento de los objetivos propuestos en la asignatura.

2.2-. Grafos: Una vez recopilados los algoritmos que intervendrán en la evaluación, se procederá a desarrollar los grafos de flujo correspondientes a cada uno de ellos. Para tal efecto se emplearán las estructuras en forma de grafo y estructuras lógicas, de tal modo de representar el algoritmo gráficamente, lo interesante de esta etapa es que se pueden obtener grafos parciales, permitiendo evaluar partes de grafos complejos.

2.3-. Matriz: Terminadas las dos etapas anteriores, se confeccionará la matriz de conexiones correspondientes al grafo de flujo. Posteriormente se reemplazarán las entradas de la matriz por 1 (unos), lo cual determinar la complejidad ciclomática de cada uno de los algoritmos.

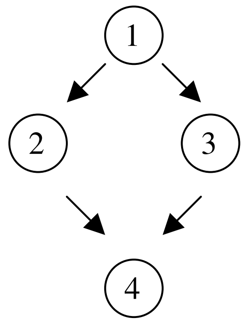
2.4-. Evaluación de complejidad: Utilizando las tres etapas anteriores, la medición de la complejidad ciclomática de un algoritmo obtenida de la matriz, nos permite clasificar la dificultad que será planteado a un curso en un período determinado. Esto se hace con el fin de cumplir de la mejor forma con los objetivos generales y específicos de la asignatura. De este modo, se puede medir de antemano la dificultad de los problemas y clasificarlos por nivel, etapa de desarrollo y curso.

3. APLICACIÓN DEL MÉTODO AGM.

A continuación mostraremos 4 ejemplos usados en la asignatura de Introducción a la Computación y Lenguajes de Programación, con una participación de 100 alumnos, distribuidos en 2 secciones, todos alumnos de la carrera de Ingeniería Civil Informática. Dichos algoritmos fueron aplicados en distintas pruebas, lo cual permitió obtener el porcentaje de alumnos que resolvieron y no lo resolvieron.

3.1-. ALGORITMO 1: Función que describe la comisión mensual por ventas ganadas por un empleado de una empresa. [3]

```
1 ←      Si venta <= 100 entonces
2 ←          Comisión = 1% * venta
          Sino
3 ←          Comisión = 1% * venta + (2% * (venta – 100))
4 ←      Fin_si
```



a) Complejidad según matriz de conexiones.

	1	2	3	4
1		1	1	
2				1
3				1
4				

Conexiones

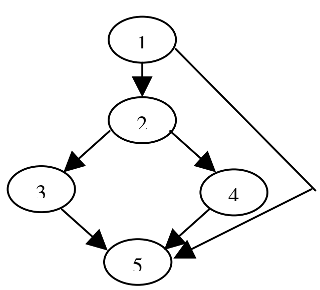
1: 2 – 1 = 1
2: 1 – 1 = 0
3: 1 – 1 = 0
4: 1 – 1 = 0

Complejidad Ciclomática V(G) = 1 + 1 = 2

b) Complejidad según regiones. N° de regiones = 2 **V(G) = 2**	c) Complejidad según aristas y nodos. V(G) = E – N + 2 V(G) = 6 – 6 + 2 **V(G) = 2**
d) Complejidad según nodos predicados. V(G) = P + 1 V(G) = 1 + 1 =2	OBS.: La complejidad ciclomática es la misma en todos los casos.

3.2-. ALGORITMO 2: Procedimiento que permite sumar dos fracciones, pasando como parámetros los dividendos y divisores de las dos fracciones y del resultado. [4]

```
1 y 2     Si (den1 <> 0 y den2 <> 0) entonces
3             num_res:=num1*den2 + num2*den1
              den_res:=den1*den2
          Sino
               Num_res:=0
4              Den_res:=0
               Escribir("División por cero")
5         Fin_si
```



a) Complejidad según matriz de conexiones.

	1	2	3	4	5
1		1			1
2			1	1	
3					1
4					1
5					

Conexiones

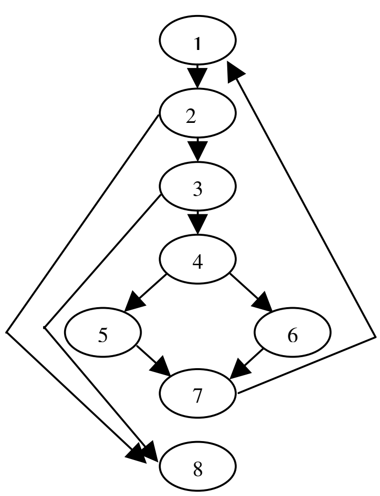
1: 2 – 1 = 1
2: 2 – 1 = 1
3: 1 – 1 = 0
4: 1 – 1 = 0
5: 1 – 1 = 0

Complejidad Ciclomática V(G) = 2 + 1 = 3

b) Complejidad según regiones. N° de regiones = 3 **V(G) = 3**	C) Complejidad según aristas y nodos. V(G) = E – N + 2 V(G) = 6 – 5 + 2 **V(G) = 3**
d) Complejidad según nodos predicados. V(G) = P + 1 V(G) = 2 + 1 = 3	OBS.: La complejidad ciclomática es la misma en todos los casos.

3.3-. ALGORITMO 3: Función que busca un elemento X dentro de un arreglo no ordenado. Retorna verdadero si existe, en caso contrario, retorna Falso. [4] [5]

```
1      i:=1
       encontrado:=false
2, 3   Mientras (i <= n) y (not Encontrado) hacer
4, 5       SI (a[i] = x) entonces Encontrado:=true
6          Sino i:=i + 1
7          Fin_SI
8      Fin_mientras
       BuscaElemento:=encontrado
```



a) Complejidad según matriz de conexiones.

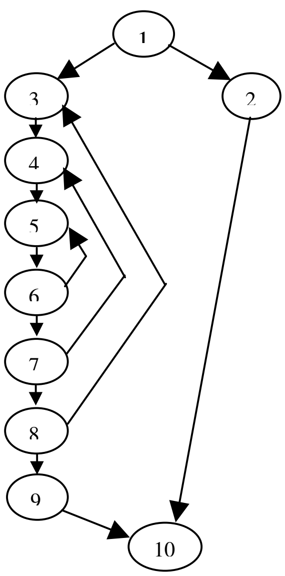
	1	2	3	4	5	6	7	8	Conexiones
1		1							1: 1 – 1 = 0
2			1					1	2: 2 – 1 = 1
3				1				1	3: 2 – 1 = 1
4					1	1			4: 2 – 1 = 1
5							1		5: 1 – 1 = 0
6							1		6: 1 – 1 = 0
7	1								5: 1 – 1 = 0
8									6: 0

Complejidad Ciclomática V(G) = 3 + 1 = 4

b) Complejidad según regiones. N° de regiones = 4 **V(G) = 4**	c) Complejidad según aristas y nodos. V(G) = E – N + 2 V(G) = 10 – 8 + 2 **V(G) = 4**
d) Complejidad según nodos predicados. V(G) = P + 1 V(G) = 3 + 1 = 4	OBS.: La complejidad ciclomática es la misma en todos los casos.

3.4-. ALGORITMO 4: Algoritmo que obtiene la matriz producto de dos matrices. [3] [4] [5]

```
1        Si N<>0 entonces
2            Error = V
         Sino Error = F
3            Desde i =1 hasta M hacer
4                Desde j = 1 hasta P hacer
                     Prod[i,j] = 0
5                    Desde k = 1 hasta N hacer
6                        Prod[i,j] = Prod[i,j] + A[i,k] * B[k,j]
7                    Fin_desde
8                Fin_desde
9            Fin_desde
10       Fin_SI
```



a) Complejidad según matriz de conexiones.

	1	2	3	4	5	6	7	8	9	10	**Conexiones**
1		1	1								1: 2 – 1 = 1
2										1	2: 1 – 1 = 0
3				1							3: 1 – 1 = 0
4					1						4: 1 – 1 = 0
5						1					5: 1 – 1 = 0
6					1		1				6: 2 – 1 = 1
7				1				1			7: 2 – 1 = 1
8			1						1		8: 2 – 1 = 1
9										1	9: 1 – 1 = 0
10											10: 0

Complejidad Ciclomática V(G) = 4 + 1 = 5

b) Complejidad según regiones. N° de regiones = 5 **V(G) = 5**	c) Complejidad según aristas y nodos. V(G) = E – N + 2 V(G) = 13 – 10 + 2 **V(G) = 5**
d) Complejidad según nodos predicados. V(G) = P + 1 V(G) = 4 + 1= 5	OBS.: La complejidad ciclomática es la misma en todos los casos.

3.5-. Tabla de resultados obtenidos

Los algoritmos presentados anteriormente formaron parte de la primera evaluación para cumplir los objetivos del curso de Introducción a la Computación y Lenguajes de Programación del año 1999, estos resultados están mostrados en base a porcentajes divididos en tres categorías: Alta, Mediana y Bajamente.

Los resultados son los siguientes:

Nºde algoritmo	Valor de la Complejidad	Complejidad Altamente lograda por los alumnos	Complejidad Medianamente lograda por los alumnos	Complejidad Bajamente lograda por los alumnos
1	**2**	**95%**	**5%**	**0%**
2	**3**	**85%**	**10%**	**5%**
3	**4**	**70%**	**20%**	**10%**
4	**5**	**15%**	**25%**	**60%**

Algunas de las lecturas de esta tabla son las siguientes: El algoritmo 1, con un tipo de complejidad 2, fue resuelto por la mayoría de los alumnos. En cambio el algoritmo 4, con una complejidad ciclomática 5 se obtiene que el 60% de los alumnos no lograron resolverlo. De lo que se deduce que a mayor complejidad el grado de logro es menor, y crece el porcentaje de algoritmos parcialmente logrados y no logrados. O bien este grado de complejidad deberían ser aplicados en evaluaciones más avanzadas de la asignatura. Por lo cual debemos aplicar un grado de complejidad adecuado según la etapa de la asignatura.

4. CONCLUSIONES.

Es importante construir instrumentos que nos permitan medir la complejidad ciclomática de un algoritmo para clasificar el grado de dificultad que será planteado a un curso en un período determinado.

El método AGM, es un instrumento que determina el nivel de un algoritmo dado, permitiendo al profesor garantizar el cumplimiento de los objetivos establecidos al inicio de la asignatura.

Existen casos en que los alumnos no responden a un determinado algoritmo propuesto en una evaluación escrita, aplicando el método AGM podemos validar las conductas no logradas por los alumnos y orientando el reforzamiento posterior.

A través del grafo que se obtiene de un algoritmo dado, se pueden establecer complejidades parciales obtenidas dentro de un caso, permitiendo obtener puntajes en la evaluación por cada complejidad parcial lograda, siendo esto de gran aporte a las pautas de evaluación de los profesores.

Si aplicamos el método AGM a las evaluaciones escritas de un curso, entonces podemos tener claramente definidas las conductas de salida, así como las de entrada de la (s) asignatura(s) posterior(es).

4.1-. Trabajos posteriores:

Uno de los trabajos posteriores, sería apoyar el método AGM con una herramienta automatizada, que permita obtener la complejidad ciclomática de un algoritmo.

Estudiar el comportamiento de la dificultad de los distintos niveles de complejidad, debido a que en este artículo se ha tratado desde un punto de vista lineal.

Otro trabajo a desarrollar es generar una equivalencia entre complejidad y costo en un proyecto de Software.

Aplicar este método AGM a OO, Lenguajes no Procedurales, Orientación a Eventos.

5. BIBLIOGRAFÍA

[1]. Roger Pressman, "Ingeniería del Software un enfoque práctico, cuarto edición, editorial Mc Graw Hill,1998.
[2]. Roger Pressman, "Ingeniería del Software: Un enfoque Práctico", tercera edición, editorial Mc Graw Hill, 1993.
[3]. Moira Castillo, "Sistema de control de Software y Manuales", Seminario de Título de DCI, Universidad Católica del Maule. 2000.
[4]. Luis Joyanes, "Fundamentos de programación, libro de problemas", editorial Mc Graw Hill1, 1996.
[5]. Luis Joyanes, "Fundamentos de programación, algoritmos y estructura de datos", editorial Mc Graw Hill, 1994.
[6]. James Antonakos, "Programación Estucturada en C", editorial Prentice Hall, 1998.