DIDÁCTICA DE OPTIMIZACIÓN VIA METAHEURÍSTICAS MEDIANTE EJEMPLOS INTERACTIVOS

ENRIQUE G. BAQUELA
Grupo de Estudios en Simulación y Optimización Industrial (GISOI)-Facultad Regional San Nicolás-Universidad Tecnológica Nacional - ARGENTINA
ebaquela@frsn.utn.edu.ar

Fecha Recepción: Noviembre 2011 - Fecha Aceptación: Agosto 2012

RESUMEN

En este artículo se plantea el uso de modelos interactivos, diseñados por los propios alumnos, para la enseñanza de optimización vía metaheurísticas.

PALABRAS CLAVES: Metaheurísticas – Optimización – Planilla de cálculo – Docencia

ABSTRACT

In this paper is proposed the use of interactive models, designed by the students, for the teaching of metaheuristics optimization algorithms.

KEYWORDS: Metaheuristics – Optimization – Spreadsheet - Teaching

1. INTRODUCCIÓN

La enseñanza de técnicas de optimización es bastante compleja desde una perspectiva netamente analítica. Es difícil para los alumnos entender el por qué del funcionamiento de un determinado algoritmo, principalmente en lo que respecta a como se obtienen las distintas soluciones a evaluar en cada solución.

Es mucho más fácil el aprendizaje si, en vez de explicarles el funcionamiento de alguna técnica, se los hace descubrirlo. Esto se puede lograr complementando el dictado habitual de la clase con herramientas que permitan una mayor interactividad en el aprendizaje. En el resto del paper veremos una aplicación interactiva en planilla de cálculo para su uso en enseñanza de metaheurísticas.

2. DESARROLLO

2.1 Conceptos

La optimización vía metaheurísticas es una rama con mucho auge de la investigación Operativa, por lo cual su enseñanza a nivel universitario se torna fundamental. Como el resto de las herramientas de IO, presenta una fuerte base matemática que, a veces, dificulta el acercamiento y/o entendimiento por parte de los alumnos al tema. Pero por suerte, las distintas metodologías de optimización tienen un equivalente geométrico que suele ser más entendible que la mera formulación algorítmica.

En gran parte de la bibliografía, vemos que el método Simplex, además de explicarse mediante su formulación con sistemas de ecuaciones, se suele introducir mediante una optimización geométrica de un sistema de dos variables de decisión. El mismo planteo puede ser hecho para el estudio de metaheurísticas.

Un enfoque que ha dado resultado en mis clases es el de enseñar, previo a los algoritmos, el concepto de espacios de soluciones. Así como con el método simplex, si entienden que todas las alternativas posibles que satisfacen un conjunto dado de restricciones forman un conjunto de puntos en un n-espacio, les resulta más fácil el entendimiento de lo que significa una búsqueda por gradiente o una selección aleatoria. Además, el trabajar sobre el espacio de soluciones tiene la ventaja de ser graficable hasta para tres variables de decisión.

El proceso algorítmico de búsqueda de la solución óptima se puede enseñar animando en un gráfico la selección y evaluación de soluciones. Se pueden tener un par de gráficos, uno que muestre las soluciones evaluadas en el espacio de soluciones y otro los valores de función objetivo hallados. Actualizando los gráficos en cada iteración de los algoritmos, podemos ver como el gráfico correspondiente a las soluciones evaluadas comienza a llenarse de a poco, y en el gráfico de valores de función objetivos vemos como en cada iteración este va evolucionando, acercándose hacia el óptimo en métodos de búsqueda basada en gradientes, aleatoriamente en métodos basados en búsquedas estocásticas. Esto permite visualizar las distintas formas de selección de soluciones a evaluar que caracterizan a cada metaheurísticas.

Si bien enseñar gráficamente y en forma animada cómo funciona el algoritmo es mucho más atractivo que enseñar el pseudocódigo de un algoritmo, se puede dar un paso más incluyendo a los alumnos en el diseño de estas representaciones gráficas. Programar una animación con un lenguaje de programación de propósito general suele ser un poco engorroso, y además no siempre los alumnos tienen los conocimientos necesarios para acometer esta tarea.

Pero resulta mucho más simple si se utiliza una planilla de cálculo, parametrizando la mayor parte del problema a resolver y el algoritmo de resolución en la misma y utilizando el lenguaje de script que suele venir embebido para la generación de la animación. Al ser esta última tarea bastante repetitiva, se puede diseñar una pequeña biblioteca para que los alumnos solo se preocupen por la parametrización del problema y del algoritmo de solución, y no por la representación gráfica. MS Excel (la planilla de cálculo más difundida) dispone del lenguaje interpretado "Visual Basic for Applications" y su alternativa FreeSoft del paquete LibreOffice permite utilizar un lenguaje similar o bien scripts en Python.

2.2 Estructura del graficador

2.2.1 Estructura del archivo de planilla de cálculo

La estructura de plantilla propuesta (que hace uso de solo una hoja de cálculo y un módulo del lenguaje de programación) para graficar el proceso de solución de un problema es la siguiente:

- La hoja de trabajo está dividida en 4 áreas:
  - Área de parámetros del problema.
  - Área de gráficos
  - Tabla de iteraciones
  - Área de cálculos del problema

- El módulo contiene los siguientes procedimientos:
  - Gestor_de_iteraciones
  - Calcular_iteración
  - Actualizar_gráficos

Estructura de la hoja de trabajo

Las cuatro secciones de la hoja de trabajo cumplen funciones bien determinadas.

La tabla de iteraciones resume toda la historia del algoritmo. Es una tabla que contiene, como mínimo, la siguiente información:

- Nro de iteración
- Solución Evaluada
- Valor Objetivo Hallado
- Satisfacción de las restricciones

La tabla de iteraciones es una tabla de valores, sin formulas, que aumenta su tamaño en una fila por cada iteración. Los datos se calculan en el área de cálculos del problema y son copiados a una nueva fila de la tabla de iteraciones.

Con esto evitamos que, si el algoritmo es complejo, el proceso requiera mucha memoria y nos genere una caída del sistema en el caso de realizar un número alto de iteraciones.

El área de cálculo del problema es la parte interesante de la planilla, ya que ella se debe codificar la estructura genérica del problema y el algoritmo de solución. Tiene más valor didáctico incluso que los gráficos, ya que aquí es donde los alumnos interactúan con el sistema y tiene que implementar el algoritmo de solución. En la misma se incluye una celda que indica si el algoritmo llegó a una solución óptima.

El área de parámetros del problema existe a efectos prácticos de no tener que recodificar el algoritmo de solución cuando cambie algún parámetro del problema. En vez de eso, el listado de parámetros se carga a una tabla y se referencias en el área de cálculos.

La última sección es el área de gráficos. En principio, incluimos dos tipos de gráficos: uno que muestre los tuplas de la solución propuesta, y otro que muestre la evolución de los valores objetivos hallados. Las referencias de los gráficos se actualizan en cada iteración, adicionando a su conjunto de datos la nueva solución evaluada, de manera tal de dar la sensación que son animados.

2.2.2 Estructura del módulo de código

Los procedimientos programados en el módulo son sencillos. Primero, el Gestor de iteraciones se encarga de guiar el proceso. Su seudocódigo es:

```
Gestor_De_Iteraciones{
        Para i= 1 Hasta Cantidad_De_Iteraciones Repetir {
                Calcular_Iteracion
                Actualizar_Grafico
                Si CeldaEncuentroOptimo = VERDADERO, AbandonarBucle
        }
}
```

El procedimiento “Calcular_Iteracion” se encarga de:

- Referenciar a las iteraciones anteriores que sea necesario para calcular la iteración actual (generalmente la última)
- Copiar el resultado de la iteración en una nueva fila en la tabla

El cálculo correspondiente a la iteración en curso es automático, ya que se encuentra parametrizado en el área de cálculos.

Por último, el procedimiento “Actualizar_Graficos” se encarga de agregar la última fila de la tabla de iteraciones al conjunto de datos de los gráficos.

2.3 Utilización en clase

Esta metodología complementa a la metodología didáctica que elija el docente, permitiendo un acercamiento práctico a la estructura de los problemas y los métodos de solución. En el caso puntual de mis clases, a fin de maximizar el entendimiento de las distintas metaheurísticas, el enfoque en el dictado de las mismas ha sido puesto en el espacio de soluciones y las diversas formas de recorrerlos, esquematizando los distintos métodos en un continuo entre el barrido de todo el espacio de solución a la búsqueda por proximidad partiendo de una solución inicial.

Se explica brevemente el algoritmo a estudiar, se esquematiza su movimiento dentro del espacio de soluciones y luego se procede al modelado en conjunto en la planilla. Los primeros problemas a optimizar suelen ser funciones simples (de manera tal que puedan determinar los valores óptimos analíticamente) o modelos que se resuelven previamente mediante programación lineal, de manera de poder evaluar la calidad del algoritmo y ver como esta es una función de los parámetros del algoritmo. Posteriormente, se les encarga modelar problemas más complejos, sin verificación previa de sus resultados.

2.4 Ejemplos de aplicación

Como primer ejemplo, se muestra un caso de optimización por búsqueda aleatoria, en un problema de maximización con dos variables de decisión sujeto a tres restricciones funcionales y cuatro restricciones de acotación (valores máximos y mínimos para cada variable). Se optó por chequear en cada iteración el cumplimiento de las restricciones, arrojando valores lógicos mediante comparación, plantear las restricciones de acotamiento en el área de parámetros y en base a ellas generar los valores de variables de decisión, y generar los valores aleatorios en el área de cálculos. Los gráficos representados corresponden al espacio de soluciones muestreado y a la evolución de la función objetivo iteración tras iteración.

Área de cálculos	
X Aleatoria	15,0299084
Y Aleatoria	38,2122234

Área de parámetros		
Cota Inferior	Cota Superior	Variable
0	50	X
0	50	Y
Iteraciones	100	

Tabla de Iteraciones

Iteración	X	Y	Restricción 01	Restricción 02	Restricción 03	Función Objetivo
1	33,8505909	0,34109422	FALSO	VERDADERO	VERDADERO	68,72446438
2	47,6193454	3,70334127	FALSO	VERDADERO	VERDADERO	106,3487146
3	12,8432733	27,2929404	FALSO	FALSO	VERDADERO	107,5653678
4	37,2449709	47,3689894	FALSO	FALSO	VERDADERO	216,59691
5	11,132692	22,1253573	FALSO	FALSO	VERDADERO	88,64145589
6	30,8948056	43,4763666	FALSO	FALSO	VERDADERO	192,2187109
7	25,8250896	44,5996063	FALSO	FALSO	VERDADERO	185,448998
8	38,9053232	36,6465721	FALSO	FALSO	VERDADERO	187,7503628
9	30,0229043	33,3887905	FALSO	FALSO	VERDADERO	160,2121802
10	26,3622035	22,4806809	FALSO	FALSO	VERDADERO	120,1664496
11	24,798573	7,0802775	FALSO	VERDADERO	VERDADERO	70,83797846
12	18,0486979	25,5598783	FALSO	FALSO	VERDADERO	112,7770308
13	37,7861724	38,1173317	FALSO	FALSO	VERDADERO	189,9243397
14	11,8547817	17,6842983	FALSO	FALSO	VERDADERO	76,76245841
15	40,7288707	26,0048887	FALSO	FALSO	VERDADERO	159,4724075
16	49,4335983	38,3490187	FALSO	FALSO	VERDADERO	213,9142527
17	23,8676562	21,8872292	FALSO	FALSO	VERDADERO	113,3969999

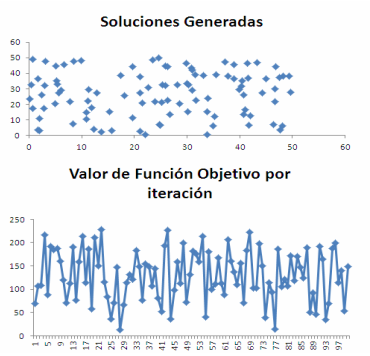
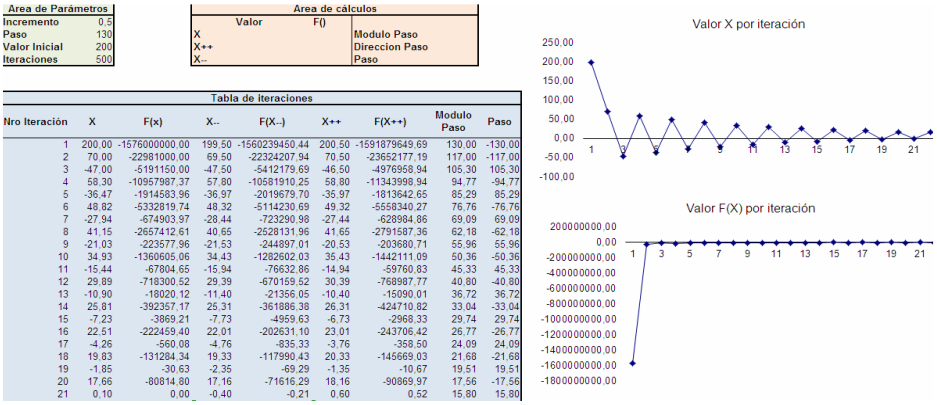


Figura 1

Como segundo ejemplo, se muestra la implementación gráfica de un algoritmo del tipo “Hill Climbing” sobre una función a optimizar de una sola variable sin restricciones funcionales. Se grafica la evolución de la selección de los valores de la variable de decisión iteración tras iteración, y los valores que toma la función objetivo también en cada iteración.



Area de Parámetros	
Incremento	0,5
Paso	130
Valor Inicial	200
Iteraciones	500

Area de cálculos		
	Valor	F()
X		Modulo Paso
X++		Direccion Paso
X--		Paso

Tabla de iteraciones

Nro Iteración	X	F(x)	X--	F(X--)	X++	F(X++)	Modulo Paso	Paso
1	200,00	-1576000000,00	199,50	-1560239450,44	200,50	-1591879649,69	130,00	-130,00
2	70,00	-22981000,00	69,50	-22324207,94	70,50	-23652177,19	117,00	-117,00
3	-47,00	-5191150,00	-47,50	-5412179,69	-46,50	-4976958,94	105,30	105,30
4	58,30	-10957987,37	57,80	-10681910,25	58,80	-11343998,94	94,77	-94,77
5	-36,47	-1914583,96	-36,97	-2019679,70	-35,97	-1813642,65	85,29	85,29
6	48,82	-5332819,74	48,32	-5114230,69	49,32	-5558340,27	76,76	-76,76
7	-27,94	-674903,97	-28,44	-723290,98	-27,44	-628984,86	69,09	69,09
8	41,15	-2657412,61	40,65	-2528131,96	41,65	-2791587,36	62,18	-62,18
9	-21,03	-223577,96	-21,53	-244897,01	-20,53	-203680,71	55,96	55,96
10	34,93	-1360605,06	34,43	-1282602,03	35,43	-1442111,09	50,36	-50,36
11	-15,44	-67804,65	-15,94	-76632,86	-14,94	-59760,83	45,33	45,33
12	29,89	-718300,52	29,39	-670159,52	30,39	-768987,77	40,80	-40,80
13	-10,90	-18020,12	-11,40	-21356,05	-10,40	-15090,01	36,72	36,72
14	25,81	-392367,17	25,31	-361886,38	26,31	-424710,82	33,04	-33,04
15	-7,23	-3869,21	-7,73	-4959,63	-6,73	-2968,33	29,74	29,74
16	22,51	-222459,40	22,01	-202631,10	23,01	-243706,42	26,77	-26,77
17	-4,26	-560,08	-4,76	-835,33	-3,76	-358,50	24,09	24,09
18	19,83	-131284,34	19,33	-117990,43	20,33	-145669,03	21,68	-21,68
19	-1,85	-30,63	-2,35	-69,29	-1,35	-10,67	19,51	19,51
20	17,66	-80814,80	17,16	-71616,29	18,16	-90869,97	17,56	-17,56
21	0,10	0,00	-0,40	-0,21	0,60	0,52	15,80	15,80

FIGURA 2

2.5 Resumen de los beneficios potenciales de la metodología

El principal beneficio de esta metodología es que, al mismo tiempo que se enseña la teoría de los algoritmos de optimización, los alumnos practican una forma genérica de implementación, siendo ellos los hacedores de los resultados de las técnicas en vez de ser solo un actor pasivo en el proceso de transmisión de conocimientos.

Como un beneficio residual, se podría resaltar el bajo costo de implementación (tanto por la no necesidad de adquirir software específico, como por el poco trabajo que requiere al docente la preparación de las plantillas).

2.6 Resultados de la aplicación de la metodología

Al estar esta nueva metodología en aplicación desde un periodo relativamente reciente no se disponen todavía de un volumen de datos suficientes para generar estadísticas fehacientes. En un trabajo posterior se compilaran las experiencias de los alumnos y se comparará el desempeño de esta metodología respecto a la asimilación de los mismos temas en ausencia de la misma.

3. CONCLUSIÓN

La complejidad de la formulación de un algoritmo de optimización no debería ser una causa suficiente para que no sea enseñado al practicante de IO.

Con el desarrollo actual de la informática, es posible diseñar herramientas didácticas que simplifiquen el proceso de aprendizaje, y que permitan tratar temas complejos en una forma más asimilable.

REFERENCIAS

- DISEÑO DE APLICACIONES PERSONALIZADAS PARA LA ENSEÑANZA EN CARRERAS DE INGENIERÍA, (2011). CALIGARIS M., RODRIGUEZ G., SCHIVO M., ROMITI M., LAUGERO L. I Jornada de enseñanza de la ingeniería (programa TEYEI).

- FÍSICA CON ORDENADOR – GARCÍA A. http://www.sc.ehu.es/sbweb/fisica/default.htm