Utilización de Software en la docencia de Técnicas Algorítmicas

I. Dorta, A. Rojas, C. Léon[1]
Dept. de Estadística, I.O. y Computación
Universidad de La Laguna
38271 La Laguna
e-mail: isadorta@ull.es

P. Dorta
Dept. de Métodos Cuantitativos en Economía y Gestión
Universidad de Las Palmas de G.C.
35017 Las Palmas
e-mail: pablo@empresariales.ulpgc.es

Resumen

La optimización combinatoria tiene una gran repercusión en los campos de planificación, producción y gestión de recursos. En este trabajo se presenta un ejemplo del tipo de problemas que se plantea a los alumnos de diversas asignaturas de Centros de las Universidades Canarias.

En primer lugar se describe el entorno donde ha tenido lugar la experiencia docente y se presentan las herramientas que se han utilizado en la realización de las prácticas: LINGO y MALLBA. A continuación, se describe con detalle una propuesta de ejercicio práctico utilizando el Problema de la Mochila Entera. Finalmente, se describen los resultados derivados de la utilización de estas herramientas en la docencia.

1. Introducción

En este trabajo se presenta un ejemplo concreto del tema de técnicas algorítmicas, tal y como se imparte en diversas asignaturas de Centros de las Universidades de Las Palmas de Gran Canaria y de La Laguna. En la realización de los ejercicios prácticos se propone el uso de dos herramientas: LINGO [3] y MALLBA [1].

LINGO es una herramienta de programación matemática que resuelve una amplia gama de problemas de optimización, lineales, no lineales y enteros, utilizando un lenguaje sencillo, lo que lo convierte en un asistente ideal en la docencia. Para resolver los problemas lineales utiliza el método del Simplex revisado, para los problemas no lineales utiliza el método generalizado del gradiente reducido y para la programación entera utiliza ramificación y acotación. Esta herramienta se usa habitualmente en la asignatura "*Matemáticas II*" (álgebra y programación lineal) que se imparte en la Licenciatura en Economía, la Licenciatura en Administración y Dirección de Empresas y la Diplomatura en Ciencias Empresariales de la Universidad de Las Palmas de Gran Canaria. Se emplea LINGO y no otros paquetes de optimización matemática (por ejemplo, LINDO) dado que el hecho de resolver problemas más generales a los lineales puede resultar de utilidad para el alumno tanto en otras asignaturas de la carrera como en su desarrollo profesional.

MALLBA es una librería para la resolución de problemas de optimización combinatoria que está siendo desarrollada en un proyecto coordinado entre la Universidad de La Laguna, la Universidad Politécnica de Cataluña y la Universidad de Málaga. El grupo de la Universidad de La Laguna tiene como objetivo el desarrollo de técnicas de resolución de problemas en el ámbito de los métodos exactos. La solución exacta de la mayoría de los problemas combinatorios implica la enumeración de los elementos de un espacio de soluciones exponencial. La búsqueda exhaustiva de una solución, dependiendo de las características del problema, se puede acelerar a través de técnicas como Divide y Vencerás, Ramificación y Acotación, y Programación Dinámica. Para estudiar la viabilidad de la versión 1.0 de la herramienta en la docencia, se ha incorporado el uso de la librería MALLBA en la docencia de la asignatura "*Herramientas y Lenguajes de Programación*" del Programa de Doctorado de Física e Informática de la Universidad de La Laguna. Puesto que la experiencia ha sido muy positiva, se ha planteado la utilización de la herramienta en otros estudios universitarios.

[1] Este trabajo ha sido parcialmente subvencionado por el proyecto CICYT-TIC1999-0754-C03

En este trabajo, se presenta un ejemplo para ilustrar cómo se usarían las herramientas LINGO y MALLBA en la docencia de "*Matemáticas II*". El problema que se ha considerado es el Problema de la Mochila 0/1. Dado que el objetivo de este trabajo es mostrar la utilidad del software en la docencia de optimización matemática, sólo se incluyen los conocimientos elementales necesarios para empezar a trabajar con ambos paquetes. En el apartado que viene a continuación se define el Problema de la Mochila Entera. En los dos epígrafes siguientes se presenta una aproximación a la resolución de dicho problema usando LINGO y MALLBA. Por último, se exponen los resultados obtenidos del uso de ambas herramientas en la docencia universitaria.

2. Definición del problema

Se puede plantear el Problema de La Mochila 0/1 de la siguiente forma:

"Se dispone de una mochila de capacidad C y de un conjunto de N objetos. Se supone que los objetos no se pueden fragmentar en trozos más pequeños, así pues, se puede decidir si se toma un objeto o si se deja, pero no se puede tomar una fracción de un objeto. Supóngase además, que el objeto k tiene beneficio p_k y peso w_k , para $k=1,2,...,N$. El problema consiste en averiguar qué objetos se han de insertar en la mochila sin exceder su capacidad, de manera que el beneficio que se obtenga sea máximo".

Su formulación como un problema de optimización es:

$$\max \quad \sum_{k=1}^{N} p_k x_k$$

$$\text{sujeto a}: \quad \sum_{k=1}^{N} w_k x_k \le C$$

$$x_k \in \{0,1\}, \quad k \in \{1,...N\}$$

3. Primera aproximación: una solución usando LINGO

En esta sección se formula el Problema de la Mochila 0/1 en LINGO y se resuelve para un caso particular. Además, se plantea y resuelve una aplicación del problema en el campo empresarial.

El problema de la mochila puede ser escrito directamente en la ventana de edición. Sin embargo, la utilización de la siguiente estructura y funciones permite aumentar de manera sencilla las dimensiones del problema.

```
MODEL:
  SETS:
    OBJETO/1..N/:BENEFICIO,PESO,X;
  ENDSETS

  DATA:
    CAPACIDAD=C;
    BENEFICIO=p1 p2 ... pN;
    PESO=     w1 w2 ... wN;
  ENDDATA

  MAX=@SUM(OBJETO(K):BENEFICIO(K)*X(K));

  @SUM(OBJETO(K):PESO(K)*X(K))<=CAPACIDAD;

  @FOR(OBJETO(K):@BIN(X(K)));
END
```

En el bloque SETS se definen las estructuras de datos. El tipo OBJETO es un array de N elementos y las variables BENEFICIO, PESO y X son de este tipo. De esta forma, BENEFICIO(K) hace referencia al beneficio del objeto k; de forma similar se interpreta PESO(K) y X(K). En el bloque DATA se asignan los valores conocidos a las constantes del modelo. El objetivo es maximizar el beneficio total, dado por la suma de los beneficios de los objetos que entran en la mochila. La restricción impone que la suma de los pesos de todos los objetos que entran en la mochila no sea superior a la capacidad de la mochila. La función @BIN indica que las variables de decisión son binarias. Los beneficios y pesos de los objetos pueden ser leídos de un fichero utilizando en la sección DATA la función @FILE(nombre_archivo).

A modo de ejemplo, supongamos que N=8, C=102, p_1=15, p_2=100, p_3=90, p_4=60, p_5=40, p_6=15, p_7=10, p_8=1, w_1=2, w_2=20, w_3=20, w_4=30, w_5=40, w_6=30, w_7=60 y w_8=10. La salida que muestra LINGO para este ejemplo es la siguiente:

```
Objective value:  280.0000
Variable         Value
   X(1)          1.000000
   X(2)          1.000000
   X(3)          1.000000
   X(4)          1.000000
   X(5)          0.000000
   X(6)          1.000000
   X(7)          0.000000
   X(8)          0.000000
```

```
Row  Slack or Surplus  Dual Price
  1      280.0000         1.000000
  2      0.000000         0.000000
```

Como puede observarse, para maximizar el beneficio total han de incluirse en la mochila los objetos etiquetados como 1, 2, 3, 4 y 6, obteniéndose así un beneficio de 280 unidades. La holgura es cero (Row 2), por lo que la capacidad de la mochila es cubierta en su totalidad. Además, el precio sombra del recurso capacidad es cero, lo que indica que el aumento de una unidad en la capacidad de la mochila no incrementa el beneficio total y, por tanto, la solución anterior seguiría siendo válida. En realidad, puede comprobarse que para C<110 la solución no varía. Sin embargo, cuando C=110 los objetos que entran en la mochila son aquellos etiquetados como 2, 3, 4 y 5, produciendo un beneficio total de 290 unidades.

A continuación se describe una aplicación práctica al campo empresarial del problema de la mochila en su versión continua [2]:

"Una empresa de transporte marítimo de mercancías posee un barco con tres bodegas cuyas capacidades son de 100 Tm. y 6000 m^3 para la primera bodega, 140 Tm. y 8000 m^3 para la segunda y 80 Tm. y 5000 m^3 para la tercera. Se desea transportar cuatro bienes de los que se dispone de 200, 100, 80 y 150 Tm., respectivamente. La siguiente tabla muestra el volumen e ingreso por tonelada métrica de cada bien.

Bienes	Volumen m^3/Tm.	Ingreso u.m./Tm.
1	70	1250
2	50	900
3	60	1000
4	75	1200

Se trata de determinar la cantidad de Tm. de cada bien que se debe transportar en cada bodega de forma que el ingreso sea máximo".

Denotando por X_{ij} a la cantidad de toneladas del bien i que se transporta en la bodega j, la formulación del problema es la siguiente.

```
MODEL:
  SETS:
    BIEN/1..4/:DISPONIBILIDAD, VOLUMEN,
               INGRESO;
    BODEGA/1..3/:CAP_P,CAP_V;
    ASIGNACION(BIEN,BODEGA):X;
  ENDSETS

  DATA:
    DISPONIBILIDAD= 200  100   80  150;
    VOLUMEN=         70   50   60   75;
    INGRESO=       1250  900 1000 1200;
    CAP_P=      100  140   80;
    CAP_V=    6000 8000 5000;
  ENDDATA

MAX=@SUM(BIEN(I):
    INGRESO(I)*@SUM(BODEGA(J):X(I,J)));

@FOR(BIEN(I):
@SUM(BODEGA(J):X(I,J))<=DISPONIBILIDAD(I));

@FOR(BODEGA(J):
@SUM(BIEN(I):X(I,J))<=CAP_P(J));

@FOR(BODEGA(J):
@SUM(BIEN(I):VOLUMEN(I)*X(I,J))<=CAP_V(J));

END
```

La siguiente salida de LINGO muestra el valor óptimo, la solución óptima, las holguras y los precios sombra asociados.

```
Objective value:   340000.0

Variable    Value  Reduced Cost
X(1,1)   78.57143     0.0000000
X(1,2)   50.00000     0.0000000
X(1,3)   71.42857     0.0000000
X(2,1)   10.00000     0.0000000
X(2,2)   90.00000     0.0000000
X(2,3)  0.0000000     0.2717972E-04
X(3,1)  0.0000000      71.42860
X(3,2)  0.0000000      71.42860
X(3,3)  0.0000000      71.42860
X(4,1)  0.0000000      139.2858
X(4,2)  0.0000000      139.2858
X(4,3)  0.0000000      139.2858

Row Slack or Surplus Dual Price
1        340000.0      1.000000
2       0.0000000     0.0000000
3       0.0000000      7.142857
4        80.00000     0.0000000
5        150.0000     0.0000000
6        11.42857     0.0000000
7       0.0000000     0.2273737E-12
8        8.571429     0.0000000
9       0.0000000      17.85714
10      0.0000000      17.85714
11      0.0000000      17.85714
```

4. Segunda aproximación: una solución usando MALLBA

La definición que se da de un *esqueleto* en el diccionario de la Lengua Española de la Real Academia es la que sigue: *Anat. conjunto de piezas duras y resistentes, por lo regular trabadas y articuladas entre sí, que da consistencia al cuerpo de los animales, sosteniendo o protegiendo sus partes blandas. Armazón que*

sostiene algo. Col., Cos. Rica, Guat., Méx.y Nic., Modelo o patrón impreso en el que se dejan blancos que se rellenan a mano. El sentido con el que se utilizará la palabra *esqueleto* en este trabajo, se acerca en gran medida a la última acepción de la definición.

Se denomina *esqueleto algorítmico* a un conjunto de procedimientos que constituyen el armazón con el que desarrollar programas para resolver un problema dado utilizando una técnica particular. Este software tiene algunos blancos que se han de rellenar para adaptarlo a la resolución de un problema concreto. En el mundo de los *esqueletos algorímicos* intervienen varios personajes:

- *Programador de Esqueletos*. Persona que diseña e implementa el esqueleto. Es quien decide el conjunto de clases que el esqueleto proporciona y cuáles son requeridas.
- *Programador de Soluciones*. Persona que adapta el problema real al esqueleto rellenando los huecos que el *Programador de Esqueletos* le solicita.
- *Usuario de Esqueletos*. Persona que utiliza el esqueleto ya relleno para obtener una solución a un determindo problema. También puede combinar más de un esqueleto.

La librería MALLBA se ha diseñado de forma que la tarea del *Programador de Soluciones* y del *Usuario de Esqueletos* sea lo más simple posible. En un esqueleto MALLBA se distinguen dos partes principales: una parte que implementa el patrón de resolución y que es completamente proporcionada por la librería, y una parte que el *Programador de Soluciones* ha de acabar de completar con las particularidades del problema a resolver y que será utilizada por el patrón de resolución. En el diseño de los esqueletos se ha utilizado la Orientación a Objetos (OO). A las ventajas, que esta filosofía aporta: modularidad, reusabilidad, modificabilidad, etc. se suma la facilidad de interpretación del esqueleto, principalmente porque hay una relación bastante intuitiva entre lo que son las "entidades" participantes en el patrón de resolución y las clases que implementan el esqueleto.

En los párrafos siguientes se define la interfaz de usuario del esqueleto de Ramificación y Acotación y se hace una instanciación para el problema de la Mochila 0/1.

4.1. Interfaz del esqueleto de Ramificación y Acotación

La Ramificación y Acotación es un método general que permite resolver un amplio rango de problemas de optimización combinatoria. La solución de un problema consiste en un vector de enteros que cumple un número de restricciones y optimiza una función objetivo. La función objetivo debe ser maximizada o minimizada. Partiendo de esta descripción, se abstrae que en un esqueleto deben representarse tanto el `Problema` como la `Solución.`

Dado que el área de soluciones contiene un número de elementos exponencial, es imposible explorar todas las soluciones en un tiempo razonable para problemas grandes. La técnica de Ramificación y Acotación intenta reducir el número de soluciones factible mediante una exploración sistemática del área de solución. Los algoritmos de Ramificación y Acotación dividen el área de solución paso por paso y calculan una cota del posible valor de aquellas soluciones que pudieran encontrarse más adelante. Si la cota muestra que cualquiera de estas soluciones tiene que ser necesariamente peor que la mejor solución hallada hasta el momento, entonces no necesitamos seguir explorando esta parte del árbol. Por lo general, el cálculo de cotas se combina con un recorrido en anchura o en profundidad. Para representar este proceso en un esqueleto, se utilizará un `Subproblema`.

Clases que ha de rellenar el Programador de Soluciones.

Las clases requeridas son las que el *Programador de Soluciones* ha de acabar de adaptar e implementar cuando instancie el esqueleto con el problema que quiere resolver. Las entidades requeridas en el esqueleto de Ramificación y Acotación son:

- La clase `Problem`: define la interfaz para representar un problema.
- La clase `SubProblem`: representa el área de soluciones no exploradas. Esta clase debe proporcionar las siguientes funcionalidades:

- *initSubProblem(pbm)*: inicializa el subproblema de partida a partir del problema original.
- *solve(pbm)*: esta función booleana devuelve cierto cuando el área de soluciones factibles a ser explorada es vacía.
- branch*(pbm, sps, sol)*: ha de proporcionar un algoritmo para dividir el área de soluciones factibles, esto implica, generar a partir del subproblema actual un subconjunto de problemas a ser explorados.
- *lower_bound(pbm,sol)*:esta función calcula una cota inferior del mejor valor de la función objetivo que podría ser obtenido para un subproblema dado.
- *upper_bound(pbm,sol)*:esta función calcula una cota superior del mejor valor de la función objetivo que podría ser obtenido para un problema dado.

- La clase Solution: define la interface para representar una solución.

Para todas estas clases se hace necesario la implementación de una interfaz general constituida por las siguientes funciones y operadores:

- Un constructor
- Un destructor
- <<, >>: Sobrecarga de los operadores de entrada/salida.

Clases que ha de instanciar el Usuario de Esqueletos.

Las clases proporcionadas son las que el *Programador de Soluciones* ha de instanciar cuando utilice un esqueleto, esto es, sólo ha de utilizarlas. Dichas clases son:

- La clase Setup: agrupa los parámetros de configuración del esqueleto. Se especifica el tipo de búsqueda: en profundidad, primero el mejor, etc.
- La clase Solver: esta clase implementa la estrategia de Ramificación y Acotación y mantiene información actualizada del estado de la exploración cuando éste se ejecuta. La ejecución se lleva a cabo mediante el método *run()*.

4.2. Problema de la mochila 0/1

La implementación con MALLBA del problema de la mochila queda de la siguiente forma:

Problem: se definen como datos del problema: el número de objetos, la capacidad de la mochila y el peso y beneficio de cada objeto.

```
requires class Problem {
    Number C,     // Capacidad
           N;     // Número de elementos
    vector<Number> p,  //beneficios
                   w;  //pesos
    inline Direction direction() const
    { return Maximize;}
    ...
}
```

Solution: se representa mediante un vector que contiene un verdadero o falso dependiendo de si el objeto pertenece o no a la solución.

```
requires class Solution {
    vector<bool> s;
    ...
}
```

SubProblem: en cada subproblema se usarán los siguientes datos: la capacidad actual, el siguiente objeto a considerar, el beneficio actual y la solución actual.

```
requires class SubProblem {
    Number CRest,   // capacidad actual
           obj,     // siguiente objeto
           profit;  // beneficio actual
    Solution sol;   // solution  actual
    ...
}
```

initSubProblem(): el primer subproblema considera toda la capacidad de la mochila, no se ha insertado ningún objeto, por lo tanto, el beneficio obtenido es nulo.

```
void SubProblem::initSubProblem (
        const Problem& pbm){
  CRest = pbm.C;
  obj = 0;
  profit = 0;
}
```

branch(): partiendo de un subproblema, genera dos nuevos, considerando la inserción o no de otro elemento.

```
void SubProblem::branch (
       const Problem& pbm,
       container<SubProblem>& subpbms){
  SubProblem spNO;
  SubProblem spYES;

  spNO = SubProblem(CRest,obj+1,profit);
  spNO.sol.update(false, sol);
  subpbms.insert(spNO);
  Number newC = CRest - pbm.w[obj];
  if (newC >= 0) {
    Number newPf = profit + pbm.p[obj];
    spYES = SubProblem(newC,obj+1,newPf);
    spYES.sol.update(true, sol);
    subpbms.insert(spYES);
  }
}
```

upper_bound(): calcula la cota superior incluyendo objetos hasta que la capacidad actual sea alcanzada, y para este último objeto, sólo se incluye la proporción de capacidad que quepa.

```
Bound SubProblem::upper_bound (
         const Problem& pbm){
  Bound upper, weigth, pft;
  Number i;

  for(i=obj,weigth=0,pft=profit;
       weigth<=CRest; i++) {
    weigth += pbm.w[i];
    pft += pbm.p[i];
  }
  i--;
  weigth -= pbm.w[i]; pft -= pbm.p[i];
  upper = pft + (Number)((pbm.p[i]*
          (CRest-  eigth))/pbm.w[i]);
  return(upper);
}
```

lower_bound(): calcula la cota inferior incluyendo los objetos hasta que la capacidad actual sea alcanzada. El último objeto que haga que se supere la capacidad no será incluido.

```
Bound SubProblem::lower_bound (
         const Problem& pbm, Solution& us){
  Bound lower, weigth, pft; Number i, tmp;
  us = sol;

  for(i = obj, weigth = 0, pft = profit;
      weigth <= CRest; i++){
    weigth += pbm.w[i];
    pft += pbm.p[i];
    us.s.push_back(true);
  }
  i--;
  weigth -= pbm.w[i]; pft -= pbm.p[i];
  us.s.pop_back();
  tmp = pbm.N - us.s.size();
  for (Number j = 0; j < tmp; j++)
    us.s.push_back(false);
  lower = pft;
  return(lower);
}
```

solve(): el problema está resuelto cuando no quedan objetos o cuando la mochila está llena.

```
bool SubProblem::solve(const Problem& pbm){
  return ((obj >= pbm.N) || (CRest <= 0));
}
```

5. Conclusiones

Se ha presentado la implementación de un problema de optimización combinatoria, concretamente el Problema de la Mochila 0/1, utilizando las herramientas LINGO y MALLBA. LINGO permite resolverlo de forma sencilla. La experiencia docente derivaba de su utilización ha puesto de manifiesto la idoneidad del empleo de dicha herramienta. En particular, su aplicación en la resolución de problemas relacionados con el campo empresarial ha permitido facilitar su asimilación por el alumnado, alcanzando un mejor nivel de comprensión y un buen grado de satisfacción por su parte. MALLBA permite entrar más en detalle en las partes que constituyen una técnica algorítmica. Esto es de gran ayuda al estudiar en profundidad las implementaciones de los algoritmos que resuelven un problema dado. La aproximación mediante esqueletos utilizada en el curso de doctorado se revela de gran utilidad docente, por ello se ha pensado extender su uso al primer y segundo ciclo. Hasta el momento, el uso de las herramientas se ha realizado por separado, es por ello que en este trabajo se propone un ejemplo de integración de las mismas para su utilización en la docencia de diversas asignaturas.

Referencias

[1] MALLBA: http://nereida.deioc.ull.es

[2] Suárez, R., Dorta, P., González, C., Andrada, J. y Hernández, J. *Problemas de Algebra Lineal para la Economía y la Empresa. Ejercicios resueltos, cuestiones y tratamiento con DERIVE y LINGO*. El Libro Técnico, 2001.

[3] User manual, *Solver Suite: LINDO, LINGO and WHAT´S BEST*, Lindo Systems Inc., 1996.