PROYECTO COMPLEJIDAD Y OPTIMIZACIÓN

JAVIER SIMÓN NARANJO HERRERA - 125529

CARLOS ANDRÉS CABRERA MATABAJOY - 1056287

ING. CARLOS ALBERTO RAMÍREZ RESTREPO

INGENIERÍA DE SISTEMAS DE INFORMACIÓN

UNIVERSIDAD DEL VALLE

2016

PROBLEMA

Se requiere transportar sacos con alimentos mediante n tipos de aviones A1, A2, . . . , An, desde un aeropuerto y arrojarse en m aldeas V1, V2, . . . , Vm, afectadas por inundaciones. Cada avión Ai puede transportar una cantidad de alimentos cij a cada aldea Vj . Así mismo, cada avión Ai puede realizar un número máximo ti de viajes, mientras que cada aldea Vj puede recibir un número máximo kj de aviones. Se desea determinar el número de viajes que deberá hacer cada avión a cada aldea de forma que se maximice la cantidad de alimento distribuido por día.[1]

La Tabla 1 muestra algunos datos de ejemplo para 3 aviones (n = 3) y 5 aldeas (m = 5). De esta manera, se tiene por ejemplo que el avión A1 puede transportar 10 unidades de alimento a la aldea V1, 8 unidades de alimento a la aldea V2, 6 unidades de alimento a la aldea V3, 9 unidades de alimento a la aldea V4 y 12 unidades de alimento a la aldea V5. Así mismo, el avión V1 puede realizar como máximo 50 viajes diarios.[1]

Aviones / Aldeas	V_1	V_2	V_3	V_4	V_5	Viajes Avión
A_1	10	8	6	9	12	50
A_2	5	3	8	4	10	90
A_3	7	9	6	10	4	60
Viajes Aldea	100	80	70	40	20	

Tabla 1: Ejemplo unidades por viaje, viajes máximos por avión y por aldea para n = 3 y m = 5.[1]

MODELO MATEMÁTICO

Se plantea para este proyecto un modelo que consiste en una tabla de n variables; es decir n aviones, n aldeas, n vuelos, formando así una matriz.

Aviones Ai/Aldeas Vj	V1	V2	V3	V.	V..	V(n-1)	Vn	Viajes Avión ti
A1	C11	C12	C13	C1.	C1..	C1(j-1)	C1j	t1
A2	C21	...					...	t2
A3	C31	...					...	t3
A.	C.1	...					...	t.
A..	C..1	...					...	t..
A...	c..1	...					...	t...
A(n-1)	C(n-i)1	...					...	t(n-1)
An	Ci1	Ci2	Ci3	ci.	ci..	Ci(j-1)	cij	tn
Viajes Aldeas Kj	K1	K2	K3	K.	K..	K(n-1)	Kn	

Variables:

-Xij es una variable que representa el vuelo que realiza un avión i hacia la aldea j, teniendo en cuenta que i y j pueden tomar valores desde 1 hasta n.

-Cij es una variable que representa el alimento que transporta el avión i a la aldea j, teniendo en cuenta que i y j pueden tomar valores desde 1 hasta n.

La función objetivo está definida por la suma del producto de todos los Xij*Cij, teniendo en cuenta que i y j pueden tomar valores desde 1 hasta n.

Max Σ Xij . Cij

-ti es una variable que representa el número de vuelos que puede realizar un avión i, teniendo en cuenta que i puede tomar valores desde 1 hasta n.

-kj es una variable que representa el número de vuelos que puede recibir una aldea j, teniendo en cuenta que j puede tomar valores desde 1 hasta n.

Restricciones:

-La suma de los vuelos de un avión a las aldeas debe ser menor o igual al número de vuelos que puede realizar ese avión.

Σ Xij $\leq$ ti

-La suma de los vuelos que llegan a una aldea debe ser menor o igual al número de aviones que puede recibir esa aldea.

Σ Xij $\leq$ kj

-Las variables Xij, ti y kj deben tomar valores positivos ya que valores negativos no tendrían sentido en este problema.

Xij,ti,kj $\geq$ 0

ENTORNO DE TRABAJO

Este proyecto fue abordado utilizando el lenguaje de programación Java[2], fue necesaria la instalación tanto del JRE(Java Runtime Environment) como del JDK(Java Development Kit). El JRE es un conjunto de utilidades que permiten la ejecución de programas Java, necesario para ejecutar este proyecto; y el JDK incluye las clases de Java, información sobre dichas clases y el compilador de Java.[3]

La codificación se realizó con la ayuda del entorno de desarrollo NetBeans[4] en su versión exclusiva para Java. Este entorno funciona sobre JRE y brinda las herramientas necesarias para hacer uso de JDK.

El problema se modeló como una función lineal con una serie de restricciones, existen diversas técnicas que permitan encontrar soluciones óptimas a problemas lineales y entre ellas se encuentra el método simplex. Este método ha sido implementado en diferentes herramientas y en el caso particular de este proyecto se utilizó la implementación de la herramienta LPSolve[5], esta herramienta está disponible en un gran número de lenguajes y es compatible con diversos sistemas operativos. El proyecto fue desarrollado utilizando versiones para Windows de las herramientas descritas anteriormente, todas ellas son multiplataforma y por lo tanto es posible modificar y ejecutar este proyecto sobre diferentes sistemas operativos.

Para la instalación de LPSolve en su versión para Java[6] se requieren dos tipos de componentes. El primero consta de unas librerías que se encargan de traducir la librería original escrita en el lenguaje C++ a código Java en tiempo de ejecución. Dichas librerías varían de acuerdo al sistema operativo y se encuentran disponibles en la página de LPSolve, para su instalación solo es necesario copiarlas al directorio de librerías del sistema operativo, en el caso de Windows el directorio es C:\Windows\system32

El otro componente necesario es un fichero .jar que debe ser agregado como librería en el proyecto, también está disponible en la página de la herramienta, este fichero es el encargado de brindar los métodos a nuestra aplicación para hacer uso de la librería LPSolve que copiamos al directorio de librerías del sistema operativo.

IMPLEMENTACIÓN

El proyecto fue desarrollado en una sola clase de Java (Principal.java), en esta clase se tienen una serie de métodos; cada uno realiza una parte del proceso para hallar la solución óptima y se tienen también unas variables, una matriz de alimentos, una matriz de vuelos, una lista maxAvionesAldea y una lista maxViajesAvion.

El método leerEntrada() es el encargado de crear una instancia de BufferedReader en la cual se va tener el archivo de entrada (entrada.txt), y de este archivo se lee cada línea de texto y se pasa a una lista de líneas. A partir de esa lista de líneas se ejecutan una serie de ciclos en los que se lee cada token de cada línea y es agregado a la correspondiente variable de la clase.

El método optimizar() hace uso de los métodos de LPSolve, primero se debe llamar al método LpS.makeLP(...) o alguno de los otros métodos para realizar una instancia de LPSolve, y a dicha instancia se le asignan las variables y restricciones, dicha asignación se hace con una listas y unos llamados al método lpS.addConstraintex(...) el cual recibe como parámetros la lista de datos y el tipo de restricción; la función objetivo se agrega con un llamado al método lpS.setObjFnex(). Después de tener la instancia y agregarle los atributos (variables, restricciones, función objetivo) se indica si se desea maximizar (setMaxim) o minimizar (setMinim) y ejecutando el método lpS.solve() se pueda hallar la solución óptima al problema, la cual puede ser mostrada accediendo a los atributos de lpS mediantes los métodos getObjective() y getVariables().

El método branchAndBound() almacena en un ArrayList los modelos y submodelos que se generan al hacer uso de optimizar(), se encarga de revisar en cada modelo si se tienen variables con valores no enteros y en caso de encontrarlos envía el modelo con la variable no entera de nuevo a optimizar pero con una nueva restricción; se almacenan los submodelos generados y se elimina el modelo original para ahorrar memoria. Cuando el método detecta que todos los modelos del array tienen variables enteras entonces compara los valores de las funciones objetivo de cada modelo y elige la mayor en este caso como solución óptima.

CONCLUSIONES

En el transcurso del desarrollo de este proyecto, se evidenció que problemas de la vida real se pueden plantear en forma lineal mediante técnicas de modelamiento y algoritmos de programación lineal como simplex y branch and bound, y que esto nos permite obtener el máximo aprovechamiento de ciertos recursos disponibles para cierta actividad, así como también maximizar o minimizar cualquier proceso o cálculo de la vida siempre y cuando sea posible de modelar como una función lineal.

Se comprobó que LPSolve es una herramienta muy completa para abordar problemas de optimización expresados en forma lineal, nos permite ingresar los datos de dos formas, la primera mediante la lectura de un archivo LP y la otra agregando columnas a una instancia de la herramienta y a partir de esos datos ingresados nos ofrece una serie de métodos que permiten encontrar la mejor solución, acceder al valor máximo o mínimo de la función objetivo, y dicha solución puede ser encontrada de acuerdo a las necesidades específicas del problema, ya sea un problema que requiera soluciones enteras, no enteras o mixtas.

Finalmente se concluye que no debe subestimarse el tiempo necesario para realizar cualquier actividad y/o proyecto y que hacerlo genera que no se logren a alcanzar los objetivos planteados ni siquiera haciendo un sobreesfuerzo en cercanías a la fecha de entrega.

REFERENCIAS

[1] Carlos Alberto Ramírez, Proyecto Final Complejidad y Optimización, 3 de mayo de 2016

https://campusvirtual.univalle.edu.co/moodle/pluginfile.php/1000661/mod_resource/content/1/Proyecto.pdf

[2] Oracle Inc, Java Platform, Standard Edition (Java SE) 8, 20 de mayo de 2016 .

http://docs.oracle.com/javase/8/

[3] Oracle Inc, ¿Cómo puedo empezar a desarrollar programas Java con Java Development Kit (JDK)? 20 de mayo de 2016

https://www.java.com/es/download/faq/develop.xml

[4] Oracle Inc, NetBeans IDE Download, 23 de mayo de 2016

https://netbeans.org/downloads/

[5] Kjell Eikland, LPSolve, 24 de mayo de 2016

https://sourceforge.net/projects/lpsolve/

[6] Kjell Eikland, Using lp_solve 5.5 in Java programs, 24 de mayo de 2016

http://lpsolve.sourceforge.net/5.5/Java/README.html