Ingeniería en Computación

Proyecto Terminal

"Una implementación gráfica de algunos problemas de programación lineal"

Desarrollado por:

Silvestre Ramírez Ramírez

205303332

Asesorado por:

Dr. Rafael López Bracho

México, D.F. Julio 2010

Contenido

Agradecimientos

Antes que nada me gustaría agradecer a todas aquellas personas que directa o in directamente formaron parte de mi formación en esta Universidad, porque a todos debo la realización de una etapa más en mi vida.

Al Dr. Rafael López Bracho, mi asesor de proyecto terminal, muchas gracias por haberme aceptado para realizar este proyecto, por ser un excelente profesor y un excelente ser humano es usted una de las personas que más admiro por su inteligencia y conocimientos, siempre será un ejemplo a seguir.

Un agradecimiento especial al Dr. Francisco Javier Zaragoza Martínez, por haberme dado la idea de este proyecto terminal, gracias también por ser un excelente profesor y principalmente gracias por haber germinado en mí el interés por los algoritmos.

A mis amigos Teresa, Antonio, Cesar, Arturo, Pepe gracias por su compañía, por los buenos y malos ratos que pasamos juntos, siempre estarán en mis pensamientos donde quiera que este.

A todos y cada uno de mis profesores que formaron parte de mi carrera, gracias por sus conocimientos, pero sobre todo gracias por sus consejos.

A la mujer de mi vida Cinthya, gracias porque nunca dejaste de creer en mí, sin ti nunca lo habría logrado.

Y, por supuesto, el agradecimiento más profundo y sentido para mi familia. Sin su apoyo, colaboración e inspiración habría sido imposible llevar a cabo esta meta. A mis padres, Josefa y Pedro, por su ejemplo de lucha y honestidad; a mis hermanas María Fernanda y Jessica por su tenacidad y superación, échenle ganas ustedes también lo pueden lograr; a mi hermano Pedro donde quiera que este, recuerda que siempre te estaremos esperando con los brazos abiertos.

A todos ellos muchas Gracias.

Silvestre

Introducción

El presente documento muestra una descripción del desarrollo y construcción del sistema "Gráficas de programación lineal" (GPL), producto del proyecto terminal "Una implementación gráfica de algunos problemas de programación lineal", el cual fue desarrollado en lenguaje C#, bajo el entorno de desarrollo *"Microsoft Visual C# 2008 Express Edition"*.

GPL va dirigido a los alumnos que cursan la UEA "Investigación de Operaciones I", en el cual se estudian algunos algoritmos de programación lineal, de tipo general como el algoritmo simplex, o particular como lo que resuelven problemas de transporte y redes. En esta aplicación se implementaron los siguientes algoritmos: el método simplex, el algoritmo de transporte, el algoritmo húngaro para el problema de asignación y el algoritmo de Ford-Fulkerson para el problema de flujo máximo en una red.

El objetivo principal de GPL, es que el alumno observe cómo evoluciona la solución de un problema paso a paso de manera gráfica, para lo cual se desarrollaron interfaces de usuario acordes a las necesidades de cada problema. Para este sistema se consideraron dos casos, el primero es que el alumno inserte su propio modelo y lo resuelva con este software, mientras que el segundo caso es abrir un ejemplo precargado y que también sea resuelto por GPL.

GPL tiene ciertos límites en cuanto a variables y restricciones se refiere, la justificación es simple, el objetivo del presente proyecto terminal no es desarrollar un software capaz de resolver problemas grandes, el objetivo como ya se mencionó es ser más didáctico y mostrar la solución paso a paso de manera gráfica, por ello para cada uno de los problemas considerados se pusieron ciertos límites los cuales se mencionan en el manual de usuario.

Finalmente se espera que este software sea de gran ayuda a los profesores y alumnos que estudien o en su caso impartan la UEA Investigación de Operaciones I.

Descripción Técnica.

Diseño del Sistema.

En términos generales, la funcionalidad GPL (Gráficas de programación lineal) se representa en el siguiente diagrama de casos de uso:

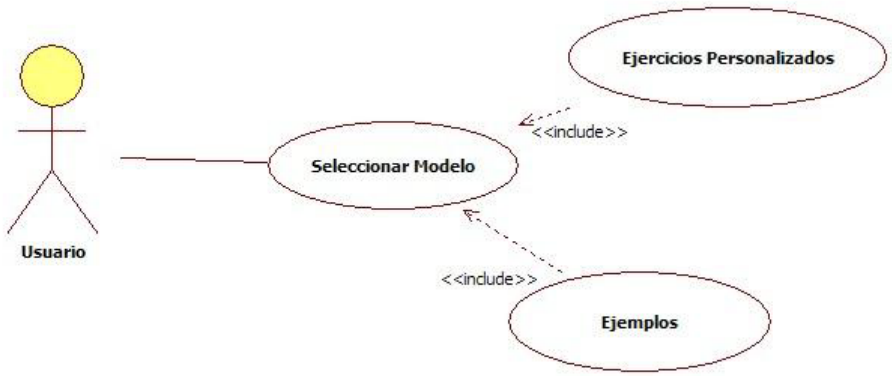


Figura 1. Diagrama de casos de uso.

Cada funcionalidad tiene los siguientes bloques:

Seleccionar Modelo

- Selección Modelo. El usuario elegirá el modelo de su interés de una lista disponible.

Ejercicios Personalizados

- **Lectura de datos.** El usuario debe introducir los datos necesarios dependiendo del modelo que seleccionó. Para el método gráfico en algoritmo simplex, el sistema acepta hasta 3 variables de decisión y 5 restricciones. Para el modelo de transporte el sistema aceptara 10 fuentes y 10 destinos, así mismo para el problema de asignación, en el cual se podrán ingresar hasta 10 trabajos y 10 máquinas. Para el problema de flujo máximo en una red se contemplan hasta 10 vértices incluidos ya el vértice Fuente y el Sumidero. La finalidad de estos límites es que no exista un amontonamiento de información en pantalla y que el alumno pueda interpretar el resultado de manera correcta. En caso de que los datos ingresados sean erróneos, el sistema lo indicará.

- **Ejecución paso a paso.** El sistema presenta en pantalla la solución intermedia de acuerdo al modelo seleccionado. Adicionalmente, en otra porción de la pantalla, se observa el paso

del algoritmo en una gráfica que puede ser en 2D o en 3D, ésta muestra la evolución de la solución a través de las distintas iteraciones por las que pasa el algoritmo.

Ejemplos

- Ejemplos prácticos. El sistema cuenta con tres ejemplos prácticos de los distintos modelos incluidos en la aplicación.

Módulos del sistema.

GPL cuenta con 4 módulos, como lo muestra la Tabla 1, dichos módulos tienen implementados los algoritmos que dan solución al problema a resolver.

Gráficas de programación lineal			
Algoritmo Simplex	Modelo de transporte	Problema de asignación	Problema de flujo máximo
	Solución inicial: esquina noroeste, costo mínimo, Vogel. Cambio de solución: Método de los multiplicadores.	Algoritmo Húngaro	Algoritmo de Ford-Fulkerson. Trayectoria Aumentante: Método de Edmonds y Karp.
Implementación gráfica R2, R3	Implementación en una gráfica bipartita.	Implementación en una gráfica bipartita	Implementación en una gráfica.

Tabla. 1 Arquitectura de la aplicación.

Algoritmos implementados.

Método Simplex.

Algoritmo Simplex.

El algoritmo Simplex, creado por el matemático norteamericano George Bernard Dantzig en 1947, es una técnica popular que da una solución numérica a un problema de la programación lineal. En general un problema de programacion lineal en su forma estandar es el siguiente.

$$\begin{array}{ll} Max\ o\ Min & Z = Cx \\ s.a & Ax = b \\ & x \geq 0 \end{array}$$

El algoritmo permite encontrar una solución óptima en un problema de maximización o minimización, buscando en los vértices de un politopo. A continuación se muestra el pseudocódigo de este algoritmo.

Función Simplex

Inicia

optimo = falso, no_acotado = falso

//Si alguna de estas variables cambia de valor el algoritmo termina

mientras (optimo == falso **y** no_acotado == falso) **hacer**

Si para todo j ($z_j - c_j \leq 0$) **Entonces**

opt = verdadero

De lo contrario inicia

Elegir alguna j tal que $z_j - c_j \leq 0$

Si $y_{ij} \leq 0$ para todo i **Entonces**

no_acotado = verdadero

De lo contrario

Encontrar $\theta_0 = \min_{i,yij>0} \left[\left[\frac{\overline{X_i}}{Y_{ij}}\right]\right] = \frac{\bar{X}}{Y_{kj}}$ y pivote es Y_{kj}

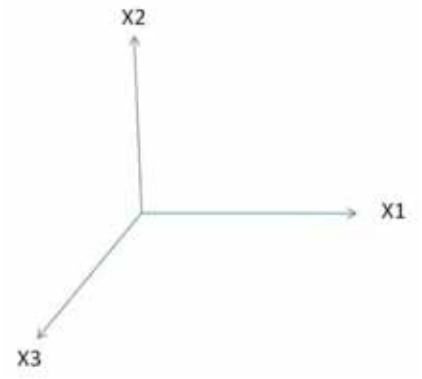
Fin

Fin

GPL tiene implementado este algoritmo, con la limitación, de sólo aceptar modelos con hasta 3 variables de decisión y 5 restricciones de desigualdad, esta decisión fue tomada debido a que sólo se puede mostrar una solución gráfica en R^3, por otro lado, el número de restricciones se debe a que el computo necesario para poder dibujar los planos es demasiado alto, finalmente se hace énfasis en que el objetivo es mostrar de manera gráfica la solución paso a paso.

Sistema de coordenadas.

A continuación se muestra el sistema de coordenadas usado para realizar las gráficas en GPL.



Sistema de coordenadas usado para las gráficas en 3D

Sistema de coordenadas para la gráfica en 2D

Interacción con la gráfica.

Para los problemas con 3 variables de decisión, en la interfaz gráfica aparece el sistema de coordenadas; así como las instrucciones para que el usuario pueda interactuar con la gráfica, ya que para poder interactuar con esta se deben presionar las teclas de dirección del teclado. A continuación se listan las teclas y sus funciones:

Tecla	Función
Arriba	Acercar
Abajo	Alejar
Derecha	Rotar gráfica a la derecha
Izquierda	Rotar gráfica a la izquierda

En la interfaz de usuario, GPL muestra el valor de las variables de decisión y su correspondiente gráfica, a continuación se muestran un par de problemas, el valor de sus variables en cada iteración y cómo evoluciona gráficamente.

Ejemplos.

Con dos variable de decisión.

Max Z $= 2X_1 + X_2$

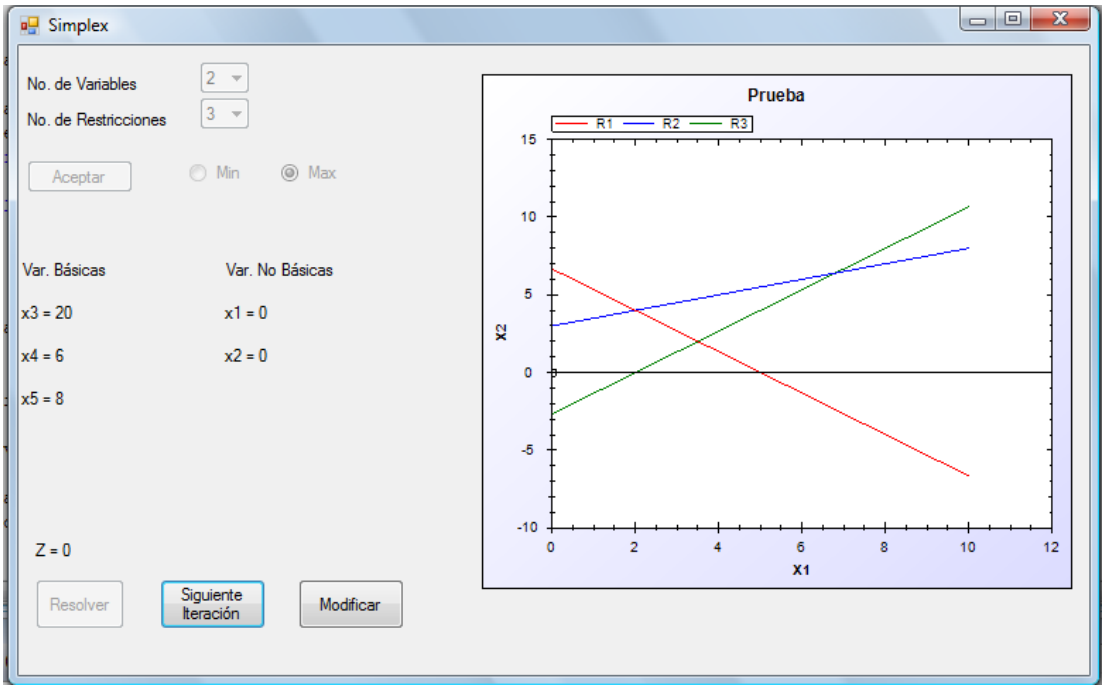
s.a $4X_1 + 3X_2 \leq 20$

$-X_1 + 2X_2 \leq 6$

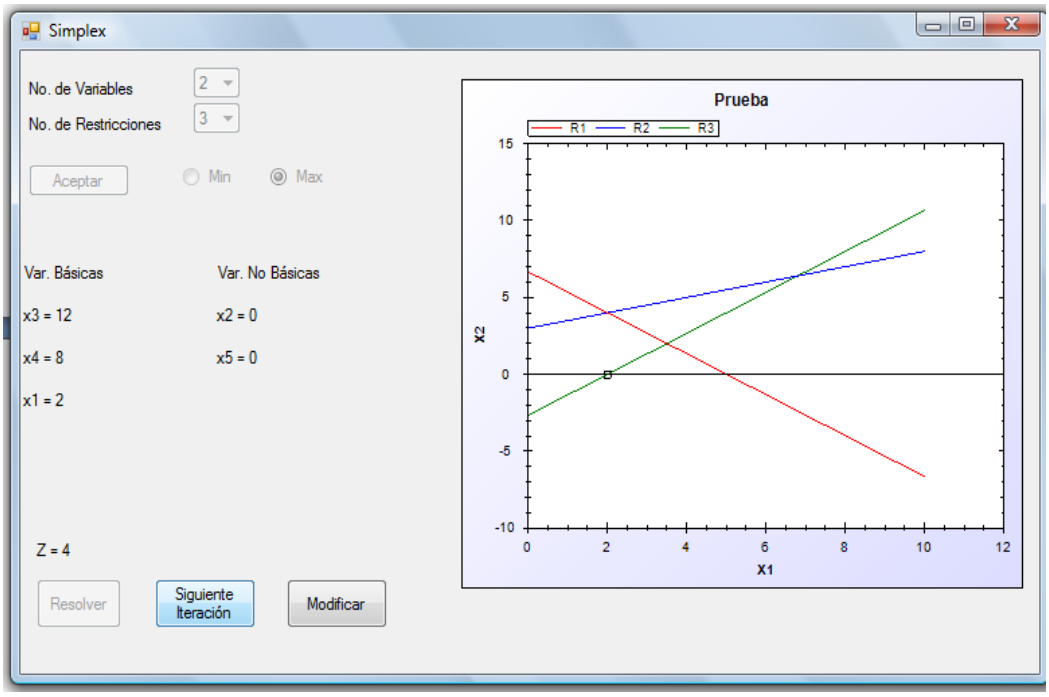
$4X_1 - 3X_2 \leq 8$

$X_1 \geq 0, X_2 \geq 0, X_3 \geq 0$

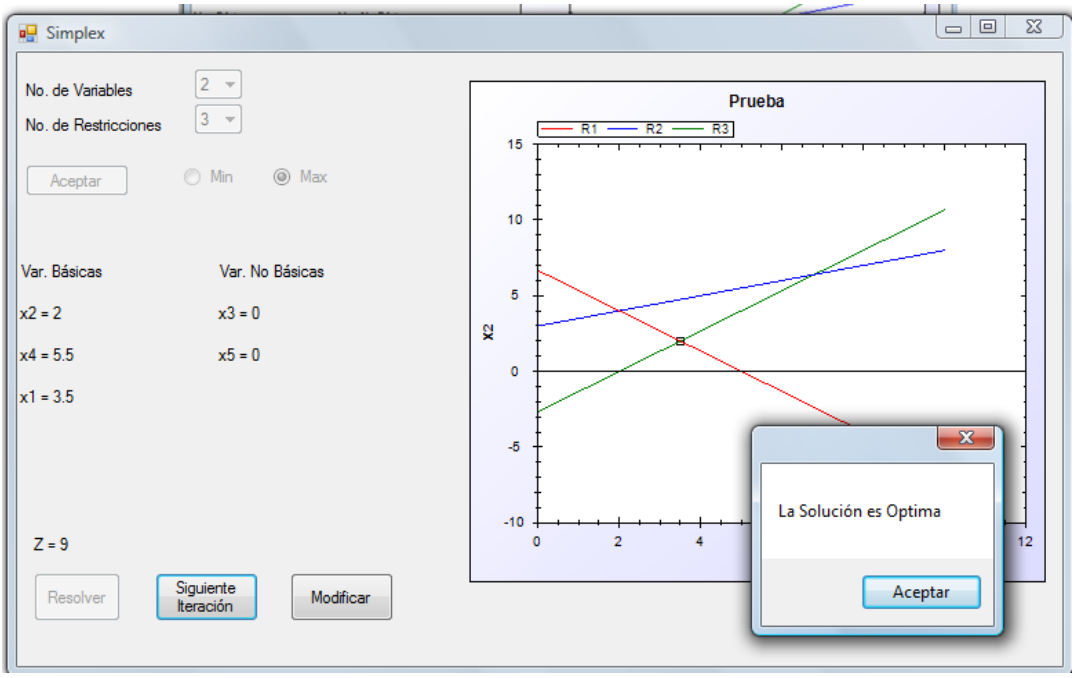
Iteración 0.



Iteración 1.



Iteración 2.



Con 3 variables de decisión.

Max Z = X_1 + X_2 + X_3

s.a X_1 + X_2 + X_3 ≤ 4

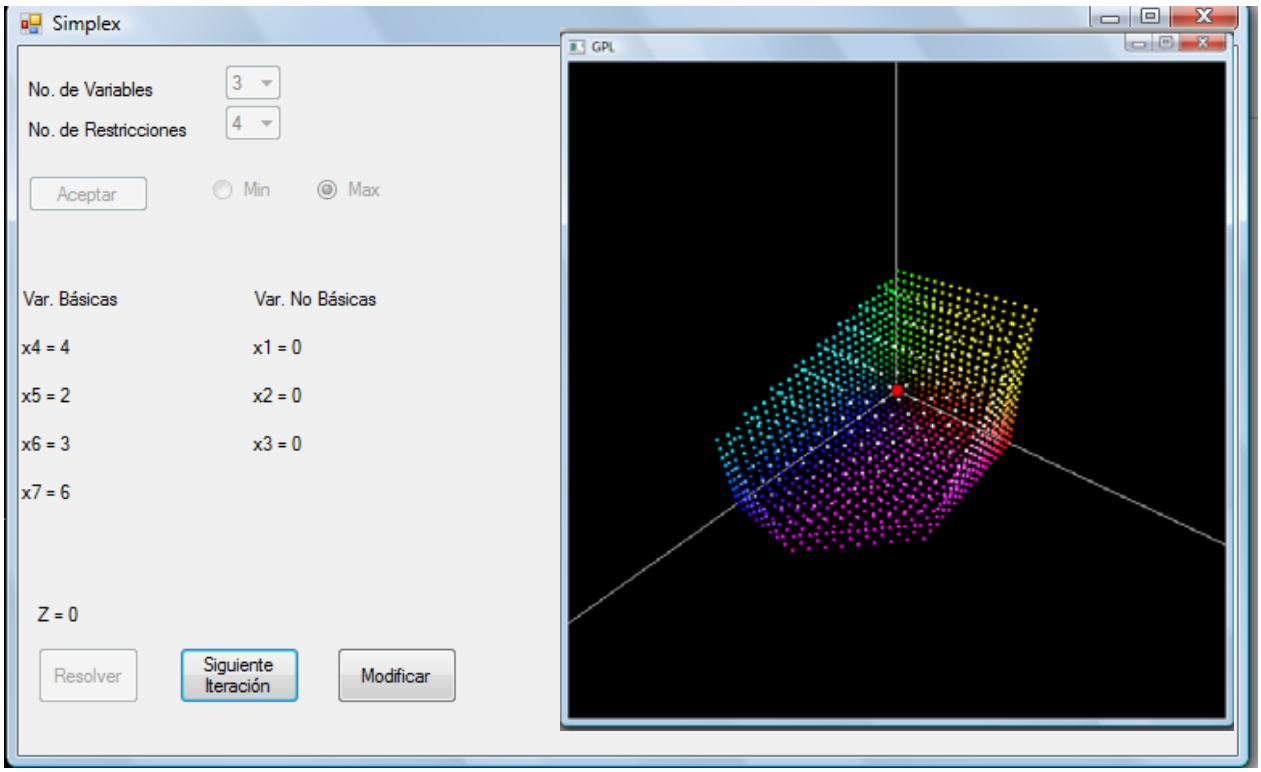
X_1 ≤ 2

X_3 ≤ 3

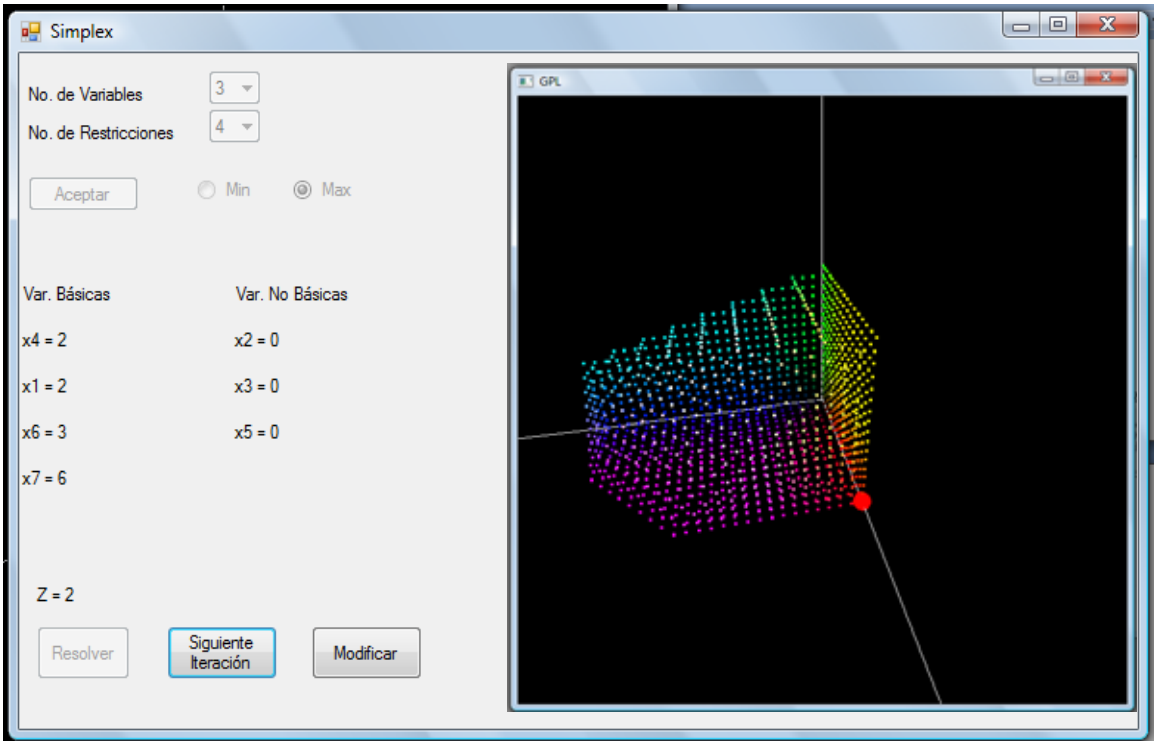
$3X_2$ + X_3 ≤ 6

$X_1 \geq 0, X_2 \geq 0, X_3 \geq 0$

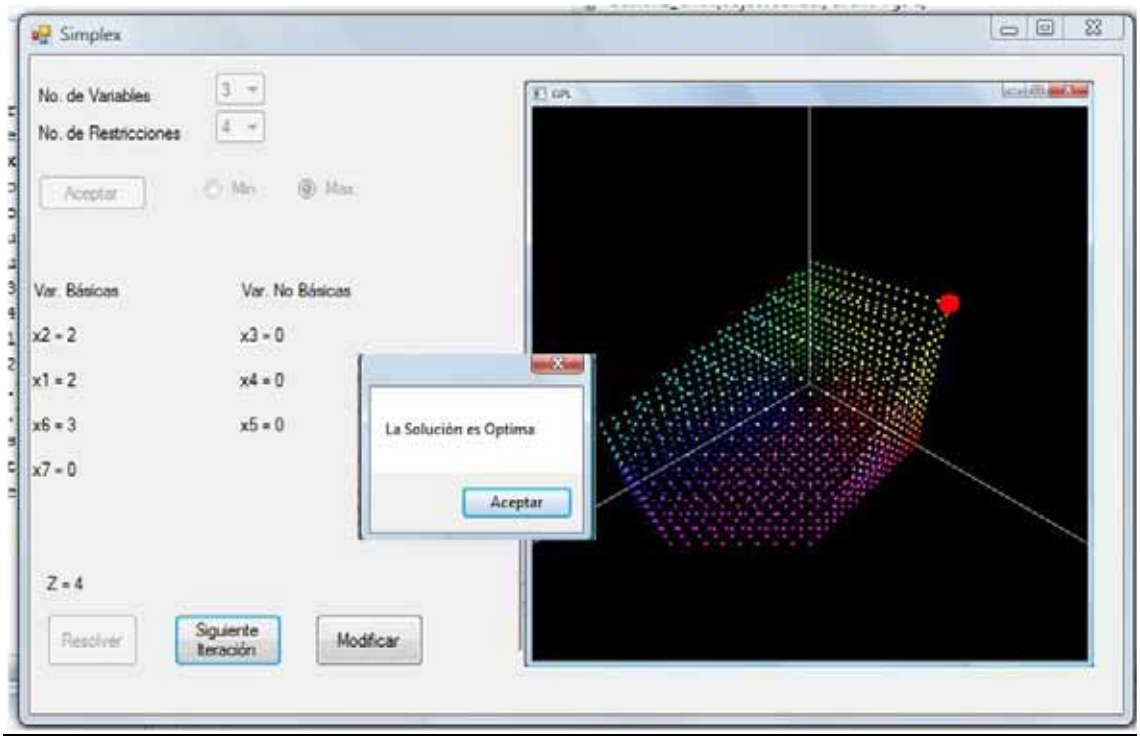
Iteración 0.



Iteración 2



Iteración 3.



Librerías usadas.

Para la realización de las gráficas mostradas anteriormente en los dos ejemplos, se usaron dos librerías distintas las cuales se mencionan a continuación:

ZedGraph: Es un conjunto de clases escritas en C# que sirve para crear graficas en 2D, es fácil de usar y tiene un alto grado de flexibilidad, fue usado para graficar las restricciones en el algoritmo simplex en 2D. ZedGraph también incluye una interfaz de "*UserControl*" (control de usuario), lo que permite arrastrar y soltar de edición en el editor de Visual Studio Forms. ZedGraph está licenciado bajo la LGPL (Licencia Pública General Reducida de GNU), es decir se puede usar libremente, así como compartirlo y modificarlo.

Tao Framework: Es una colección de librerías que facilitan el desarrollo de gráficos en 3D, tiene distintas funciones entre ellas la realización de videojuegos en la plataforma .NET de Microsoft, fue usada para la realización de las gráficas en 3 dimensiones y sólo fue usada la librería que se enlaza con Opengl y GLUT. Cabe mencionar que la implementación de la gráfica en 3D es una aplicación independiente a GPL, y es llamada desde GPL pasándole los parámetros necesarios para que pueda dibujar una escena en 3D.

GLPK: Kit de Programación Lineal GNU, aunque en un principio se tenia contemplado el uso de este paquete, no se pudo integrar a GPL, ya que para cumplir con el propósito de este proyecto terminal era indispensable que se pudiera tener acceso a cada unas de las iteraciones que hace el algoritmo Simplex, GLPK no cuenta con este requisito y era necesario modificar su código fuente por lo cual se decidió implementar un algoritmo propio, el cual tiene la ventaja de que se puede tener acceso a todos las variables y elementos. Por el momento se implementó el algoritmo con restricciones <=, pero en un futuro podrán integrarse restricciones de otro tipo.

Modelo de transporte.

El modelo de transporte se define como una técnica que determina un programa de transporte de productos o mercancías desde unas fuentes hasta los diferentes destinos al menor costo posible. Este es un modelo particular de problema de programación lineal, el cual su resolución a través del método simplex es tedioso, pero que debido a sus características especiales, nos permite desarrollar un método más práctico de solución. Matemáticamente tenemos lo siguiente:

$$Min\ Z = \sum_{i=1}^{m}\sum_{j=1}^{n} C_{ij}X_{ij}$$

$$s.a$$

$$\sum_{j=1}^{n} X_{ij} = a_i \ ; i = 1,2,\dots,m$$

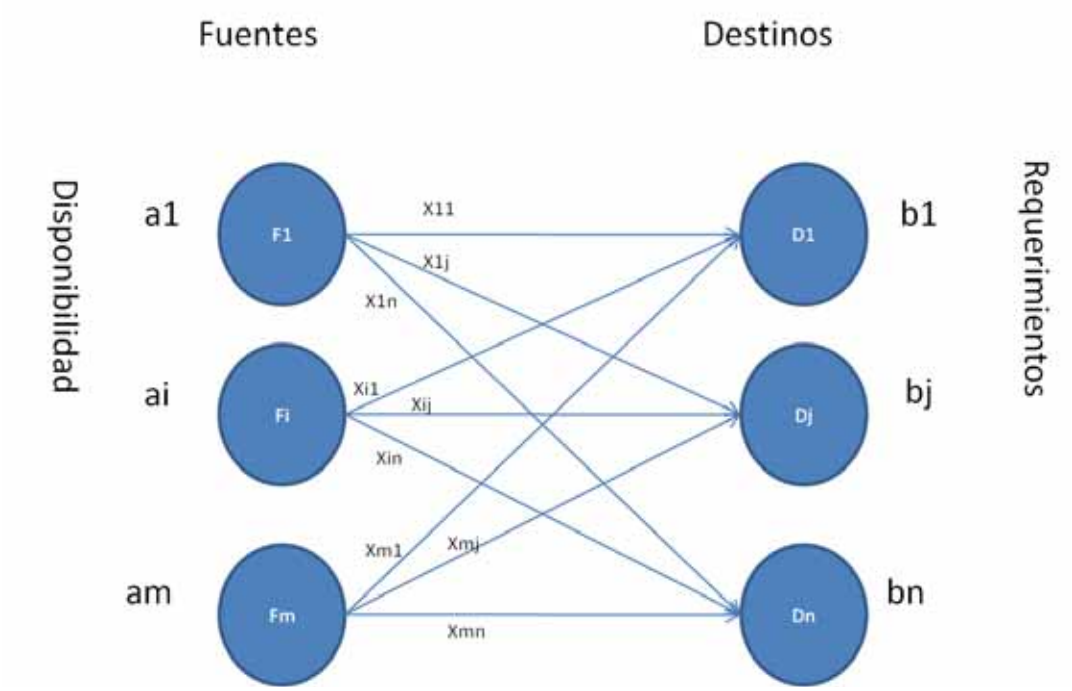
$$\sum_{i=1}^{m} X_{ij} = b_j \ ; j = 1,2,\dots,n$$

$$X_{ij} \geq 0; i = 1,\dots,m; j = 1,\dots n$$

Donde

- $X_{i,j}$= Unidades a enviar desde la fuente i-ésima (i=1,...,m) al destino j-ésimo (j=1,...,n)
- $C_{i,j}$= Costo de enviar una unidad desde la fuente i-ésima (i=1,...,m) al destino j-ésimo (j=1,...,n)
- a_i = Disponibilidad (oferta) en unidades, de la fuente i-ésima (i=1,...,m)
- b_j = Requerimiento (demanda) en unidades, del destino j-ésimo (j=1,...,n)

En este modelo se supone que todos los productos transportados han sido solicitados, esto es, que la oferta es igual a la demanda. En caso de no cumplirse este requisito, se iguala la oferta y la demanda añadiendo fuentes o destinos ficticios según sea el caso, pero cuyo tratamiento no es el objetivo de GPL, Gráficamente tenemos lo siguiente:



La metodología general para resolver un problema de transporte es el siguiente:

Para obtener una solución inicial existen distintos algoritmos, en este proyecto se emplearon tres de ellos que son los siguientes:

- Esquina noroeste
- Costo mínimo
- Aproximación de Vogel

Posteriormente se evalúa la solución y si resulta óptima el algoritmo termina, si no se usa el método de los multiplicadores para generar una nueva solución y se vuelve a probar si es óptima, se repite este paso hasta que la solución sea óptima. A continuación se explica más a detalle los métodos de solución inicial.

Solución Básica Factible Inicial.

Esquina noroeste.

Características

- Sencillo y fácil de hacer
- No tiene en cuenta los costos para hacer las asignaciones
- Generalmente proporciona una solución lejos del óptimo

Algoritmo

1. Construya una tabla de ofertas (disponibilidades) y demandas (requerimientos).
2. Empiece por la esquina noroeste (Superior Izquierda).
3. Asigne lo máximo posible (Lo menor entre la oferta y la demanda, respectivamente)
4. Actualice la oferta y la demanda y rellene con ceros el resto de casillas (Filas ó Columnas) en donde la oferta ó la demanda ha quedado satisfecha.
5. Muévase a la derecha o hacia abajo, según ha quedado disponibilidad para asignar.
6. Repita los pasos del 3 al 5 sucesivamente hasta llegar a la esquina inferior derecha en la que se elimina fila y columna al mismo tiempo.

Nota.

No elimine fila y columna al mismo tiempo, a no ser que sea la última casilla. El romper ésta regla ocasionará una solución en donde el número de variables básicas sea menor a m+n-1, no obteniendo una solución básica factible.

Costo mínimo.

Características

- Es más elaborado que el método de la esquina noroeste
- Tiene en cuenta los costos para hacer las asignaciones
- Generalmente nos deja poco alejados del óptimo

Algoritmo

1. Construya una tabla de disponibilidades, requerimientos y costos
2. Empiece en la casilla que tenga el menor costo de toda la tabla, si hay empate, escoja arbitrariamente (Cualquiera de los empatados).
3. Asigne lo máximo posible entre la disponibilidad y el requerimiento (El menor de los dos).
4. Rellene con ceros (0) la fila o columna satisfecha y actualice la disponibilidad y el requerimiento, restándoles lo asignado.
5. Muévase a la casilla con el costo mínimo de la tabla resultante (Sin tener en cuenta la fila o columna satisfecha).

6. Regrese a los puntos 3, 4, 5 sucesivamente, hasta que todas las casillas queden asignadas.

Aproximación de Vogel.

Características

- Es más elaborado que los anteriores, más técnico y dispendioso.
- Tiene en cuenta los costos, las ofertas y las demandas para hacer las asignaciones.
- Generalmente nos deja cerca al óptimo.

Algoritmo

1. Construir una tabla de disponibilidades (ofertas), requerimientos (demanda) y costos.
2. Calcular la diferencia entre el costo más pequeño y el segundo costo más pequeño, para cada fila y para cada columna.
3. Escoger entre las filas y columnas, la que tenga la mayor diferencia (en caso de empate, decida arbitrariamente).
4. Asigne lo máximo posible en la casilla con menor costo en la fila o columna escogida en el punto 3.
5. Asigne cero (0) a las otras casillas de la fila o columna donde la disponibilidad ó el requerimiento quede satisfecho.
6. Repita los pasos del 2 al 5, sin tener en cuenta la(s) fila(s) y/o columna(s) satisfechas, hasta que todas las casillas queden asignadas.

Método de los multiplicadores.

Variable entrante.

A partir de cualquiera de las soluciones iniciales anteriores, se procede a buscar una solución óptima, el método que fue usado durante el desarrollo de GPL fue el método de los multiplicadores. En el método de multiplicadores asociamos los multiplicadores U_i y V_j con el renglón i y la columna j de la tabla de transporte.

Para cada variable básica X_{ij} en la solución actual, los multiplicadores U_i y V_j deben satisfacer la ecuación que sigue:

$C_{ij} - U_i - V_j = 0$, para cada variable básica.

y $C_{ij} - U_i - V_j >= 0$, para cada variable no básica.

Estas ecuaciones producen m+n-1 ecuaciones con m+n incógnitas. Los valores de los multiplicadores se pueden determinar a partir de estas ecuaciones suponiendo un valor arbitrario para cualquiera de los multiplicadores y resolviendo las m+n-1 ecuaciones.

Posteriormente se calcula los costos reducidos $\bar{C}_{ij} = C_{ij} - U_i - V_j$ para las variables no básicas, y elegimos a aquella que tenga el costo $\bar{C}_{ij}$ más negativo (debido a que estamos en un problema de minimización), ya que ésta hace que Z disminuya más rápido.

En caso de que todos los costos reducidos $\bar{C}_{ij} > 0$, la solución es óptima.

Variable Saliente.

Este paso es equivalente a aplicar la condición de factibilidad del método simplex. Para el fin de determinar la variable que sale, construimos un ciclo cerrado para la variable actual que entra. El ciclo empieza y termina en la variable no básica designada como entrante, y se van colocando signos + y – en forma alterna comenzando por la celda de la variable que entra. Este ciclo consta de los segmentos sucesivos horizontales y verticales cuyos puntos extremos deben de ser variables básicas, de estas se elige aquella que tenga la menor asignación, posteriormente se suma y resta de los elementos del ciclo comenzando en la variable entrante.

Para la implementación de esta parte, se usó el algoritmo de Dijkstra para encontrar un camino en la gráfica, lo cual se logró haciendo uso del tablero de transporte, debido a que este esencialmente es la representación de una gráfica bipartita, bastó con implementar el algoritmo de Dijkstra para una matriz, obteniendo los resultados esperados.

Finalmente se procede a calcular nuevamente los costos reducidos C_{ij} y si estos son mayores que cero, entonces se ha llegado a una solución óptima, en caso contrario se procede nuevamente a elegir una variable que entre, la cual será aquella que cuente con el costo $\bar{C}_{ij}$ mas negativo.

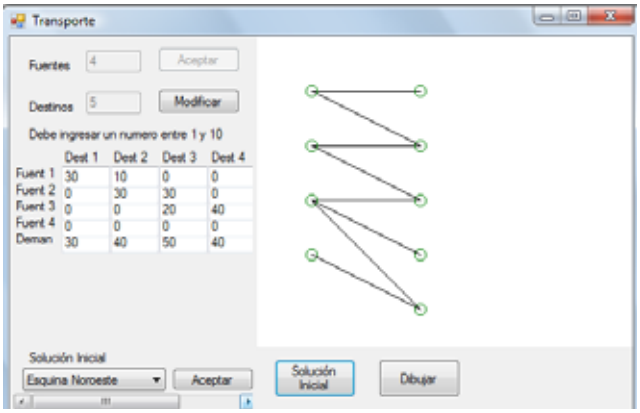
Ejemplo

Considere la siguiente tabla de transporte

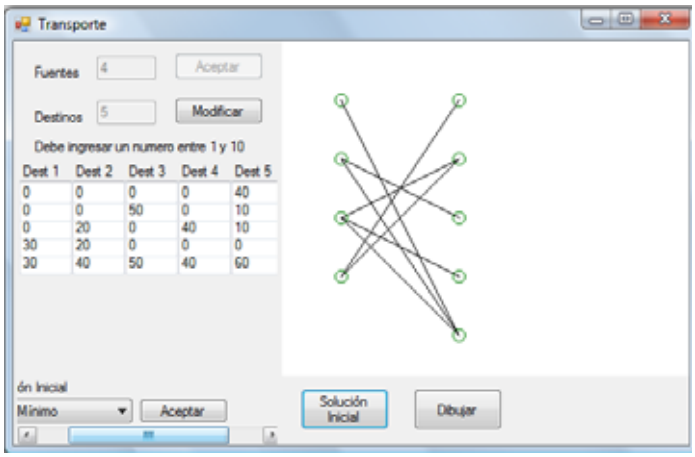
		DESTINOS					
		1	2	3	4	5	O
F	1	20	19	14	21	16	**40**
U	2	15	20	13	19	16	**60**
E	3	18	15	18	20	100	**70**
N	4	10	10	10	10	10	**50**
T	D	**30**	**40**	**50**	**40**	**60**	

Las soluciones que se obtiene por los distintos métodos son los siguientes.

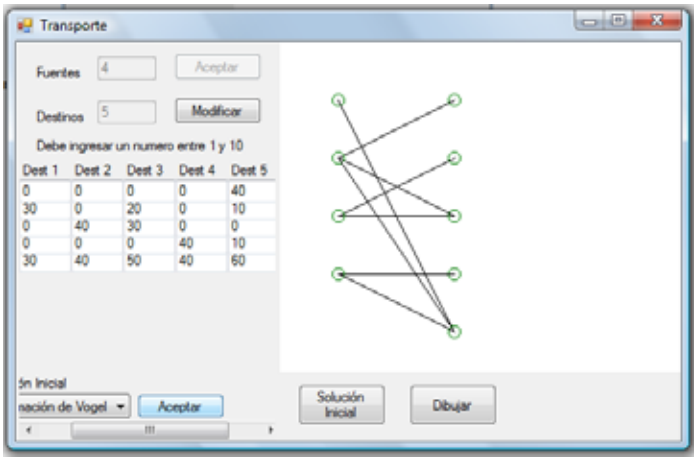
Esquina Noroeste



Costo mínimo.

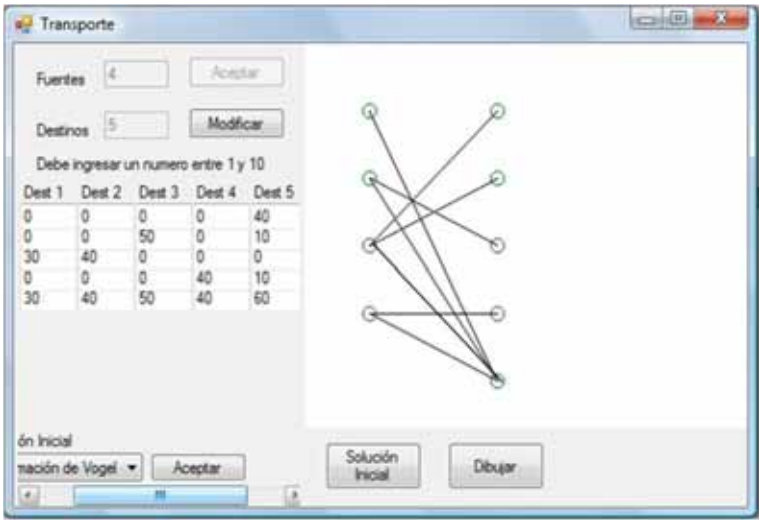


Aproximación de Vogel.



La solución óptima.

Esta solución fue obtenida con la solución inicial de Vogel y se obtuvo en una sola iteración.



Limites

En el modelo de transporte GPL está restringido a aceptar hasta 10 fuentes y destinos, la razón se ha explicado y es que no exista el amontonamiento en la gráfica, GPL tiene contemplado los casos de balanceo entre oferta y demanda. Por un lado si la oferta excede la demanda GPL informa al usuario y genera una nueva tabla en el cual incluye un destino ficticio. En el caso de que la demanda exceda a la oferta, GPL informa que no existe una solución para el problema de transporte planteado.

Problema de asignación.

El problema de asignación presenta una estructura similar a la de transporte, pero con dos diferencias: asocian igual número de orígenes con igual número de destinos y las ofertas en cada origen es de valor uno, como lo es la demanda en cada destino.

En el problema de asignación se tienen tareas y maquinas, cada tarea debe asignarse a una y sólo a una máquina. Cada máquina realizará una y sólo una tarea. El problema es por tanto, decidir el modo en el que deben realizarse todas las asignaciones para minimizar los costos totales.

$$Min\ Z = \sum_{i=1}^{m}\sum_{j=1}^{m} C_{ij}X_{ij}$$

$$s.a$$

$$\sum_{j=1}^{m} X_{ij} = 1\ ; i = 1,2,\dots,m$$

$$\sum_{i=1}^{m} X_{ij} = 1\ ; j = 1,2,\dots,m$$

$$X_{ij} \in \{0,1\}$$

La condición necesaria y suficiente para que este tipo de problemas tenga solución, es que se encuentre balanceado, es decir, que los trabajos totales sean iguales a las maquinas totales, además esto garantiza que en la solución óptima las variables tengan valores cero o uno.

Algoritmo Húngaro.

El problema de asignación puede ser resuelto mediante al algoritmo húngaro, el cual se explica a continuación.

1. Empiece por encontrar el elemento más pequeño en cada renglón de la matriz de costos. Construya una nueva matriz, al restar de cada costo, el costo mínimo de su renglón. Encuentre, para esta nueva matriz el costo mínimo en cada columna. Construya una nueva matriz (la matriz de costos reducidos) al restar de cada costo el costo mínimo de su columna.
2. Dibuje el mínimo número de líneas (horizontales o verticales) que se necesitan para cubrir todos los ceros en la matriz de costos reducidos. Si se requieren menos de m líneas para cubrir todos los ceros, siga con el paso 3, en caso contrario la solución es optima y puede

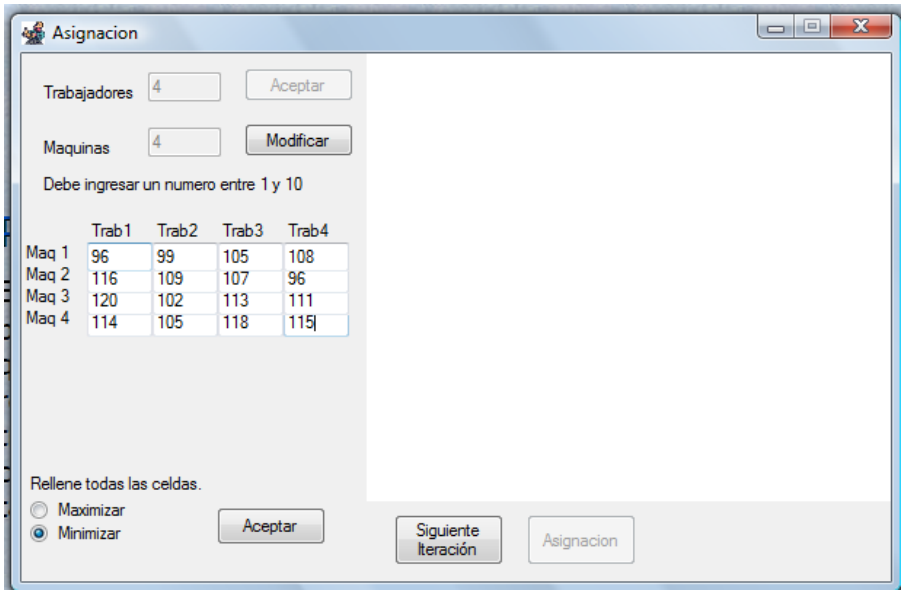
hacer una asignación óptima, seleccionando los elementos “ceros”, cuidando no repetir una asignación a una trabajador o maquina asignada anteriormente.

3. Encuentre el menor elemento no cero (llame su valor k en la matriz de costos reducidos, que no está cubierto por las líneas dibujadas en el paso 2. Ahora reste k de cada elemento no cubierto de la matriz de costos reducidos y sume k a cada elemento de la matriz de costos reducidos cubierto por dos líneas. Regrese al paso 2.

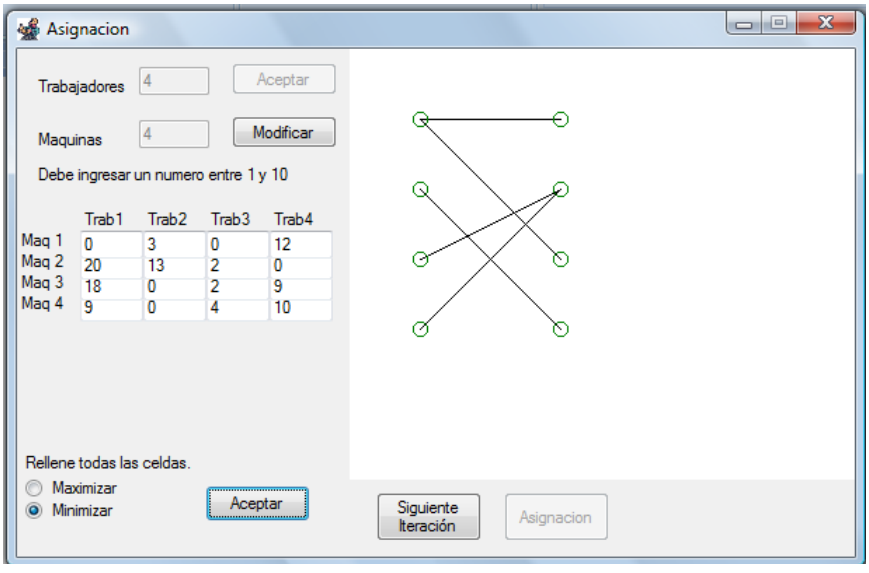
Ejemplo.

En esta aplicación se desarrollo este algoritmo para obtener la solución final y la correspondiente asignación. A continuación una secuencia de un ejemplo en GPL.

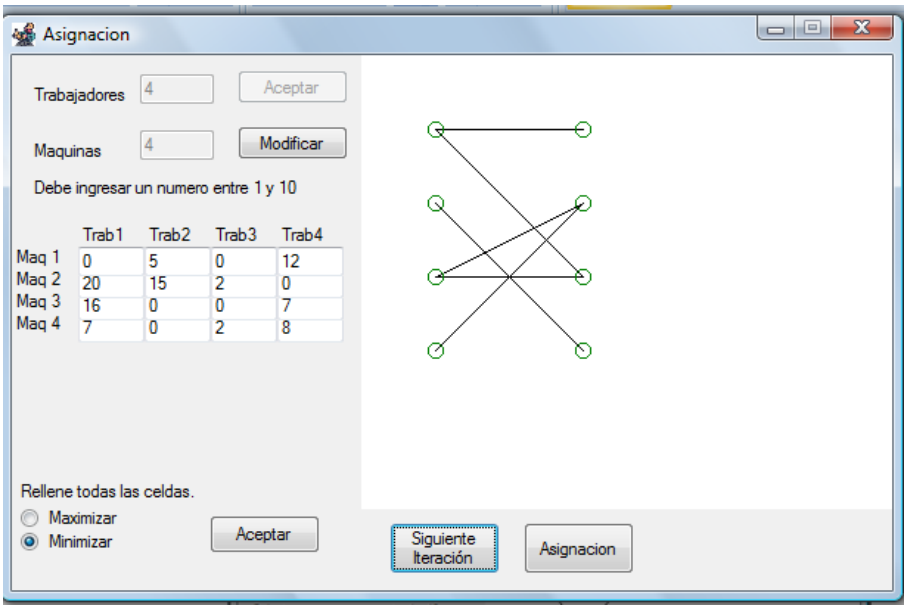
En un principio se carga el problema y no es posible hacer asignación alguna



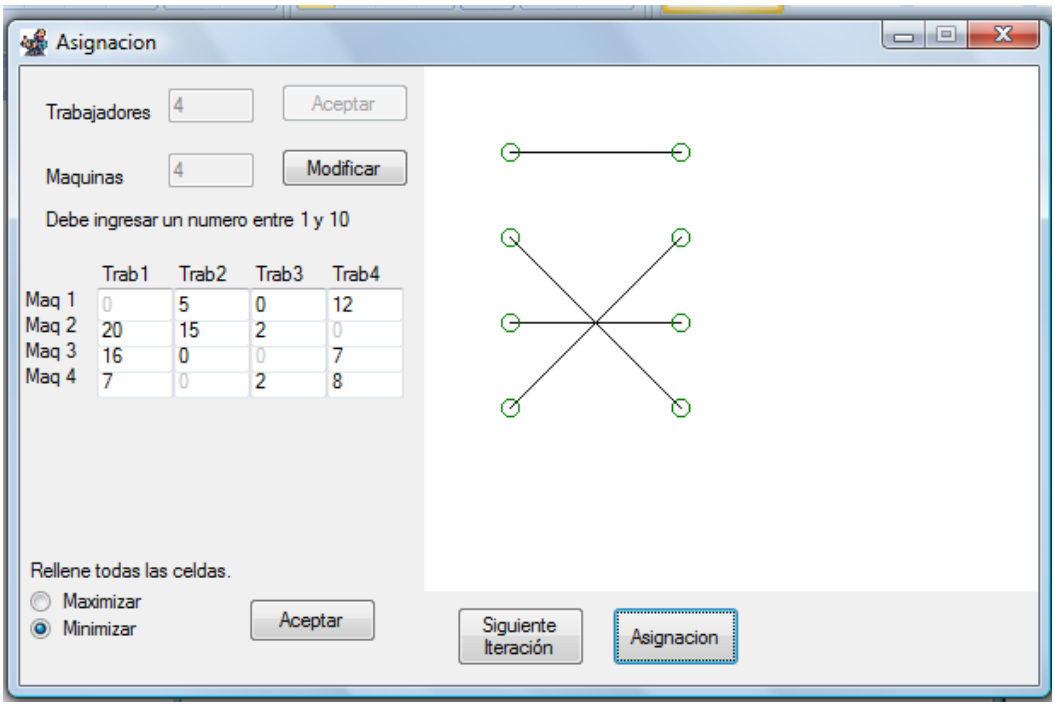
Obtenemos una solución inicial, restando de cada fila el elemento más pequeño de cada fila, y posteriormente restando el elemento más pequeño de cada columna de la matriz resultante.



En una nueva iteración obtenemos la solución óptima.



Presionamos el botón asignación para obtener la asignación óptima.



Problema de flujo máximo.

Una red de flujo es un gráfica dirigida G=(V,A) en donde cada arco (u,v)∈A tiene una capacidad no negativa c(u,v)≥0.

- Se distinguen dos vértices: la Fuente F y el Sumidero S.
- Se asume que cada vértice se encuentra en alguna ruta de F a S.
- El flujo que se envíe por cada arco no debe exceder su capacidad.
- El flujo que sale de F es el mismo que entra a S.

El problema de flujo máximo trata de maximizar este flujo de F a S.

Algoritmo de Ford-Fulkerson.

La solución al problema de flujo la da el algoritmo de Ford - Fulkerson, el cual se enuncia a continuación:

Ford-Fulkerson (G,s,t)

inicializar flujo f a 0;

while (existe un camino p de aumento F a S) do

aumentar el flujo f a lo largo de p;

return f;

La idea general es encontrar un camino de F a S, si este existe, buscar la capacidad mínima de los arcos que conforman el camino y aumentar el flujo de F a S. Cuando ya no exista camino alguno. GPL tiene implementado este algoritmo, además para encontrar un camino aumentante se implementó la búsqueda a lo ancho, propuestá por Edmonds y Karp.

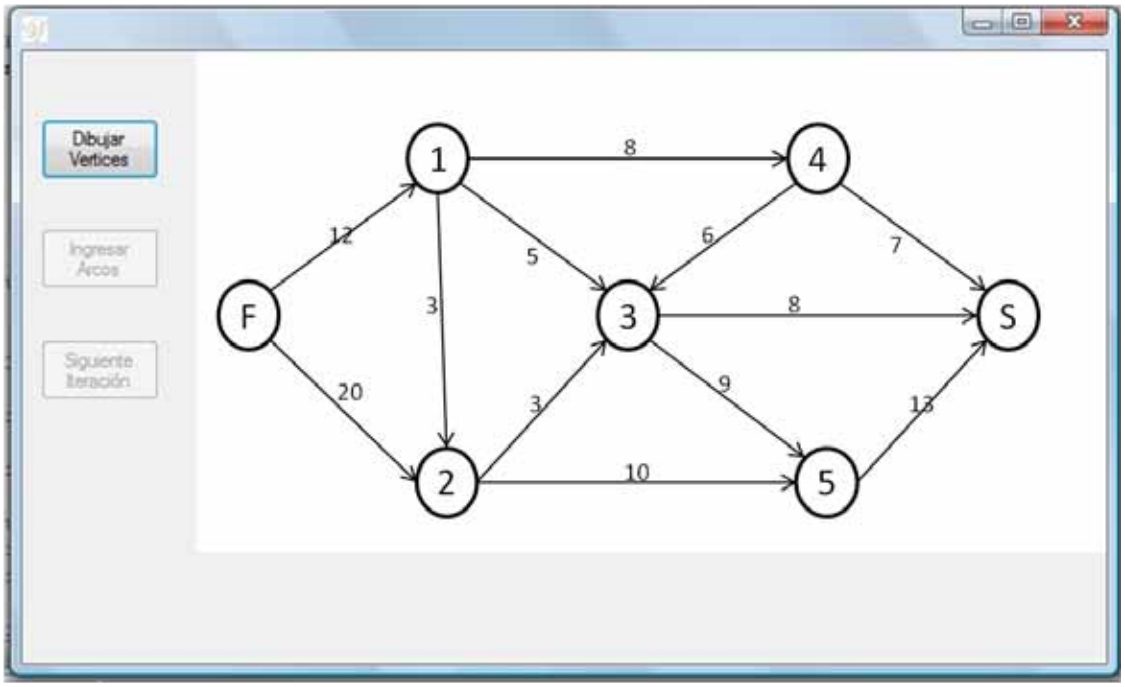
GLP permite diseñar un gráfica personalizadá, la solución se basa como ya se mencionó en el algoritmo de Ford-Fulkerson, mostrando en la aplicación la trayectoria aumentante y el flujo seleccionado. A continuación un ejemplo, la gráfica se obtiene de la siguiente matriz de adyacencia.

Ejemplo

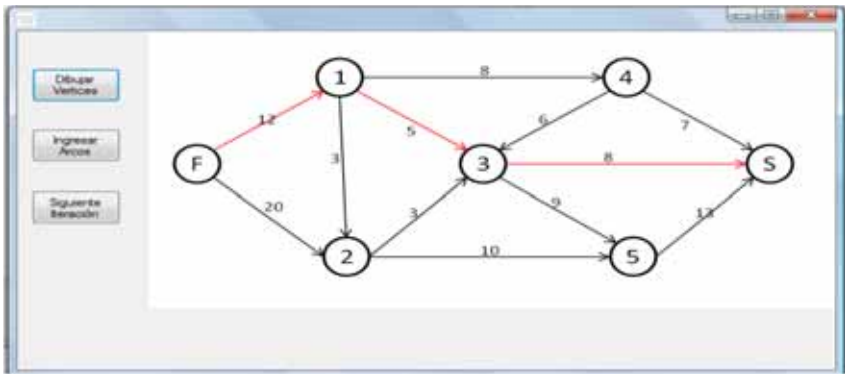
La siguiente matriz de adyacencia representa una gráfica, el usuario debe ingresar esta información para que se le genere la gráfica con esta información.

	F	1	2	3	4	5	S
F	0	12	20	0	0	0	0
1	0	0	3	5	8	0	0
2	0	0	0	3	0	10	0
3	0	0	0	0	0	9	8
4	0	0	0	6	0	0	7
5	0	0	0	0	0	0	13
S	0	0	0	0	0	0	0

La gráfica generada es la siguiente:

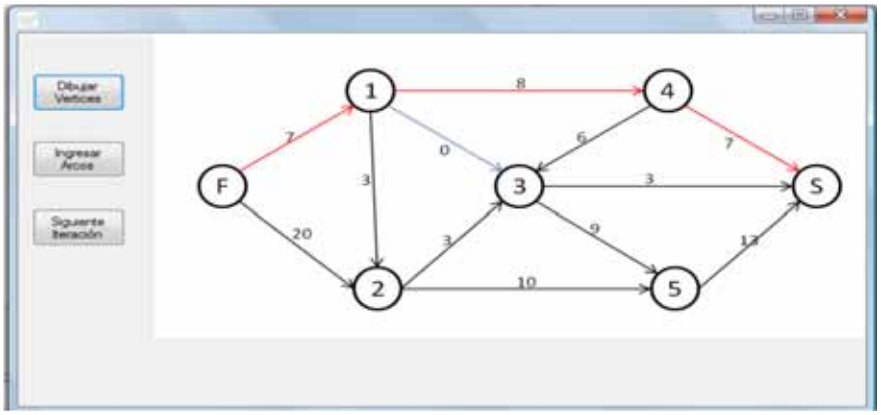


Obtenemos la primer trayectoria aumentante F-1-3-S, donde el máximo flujo posible es 5.

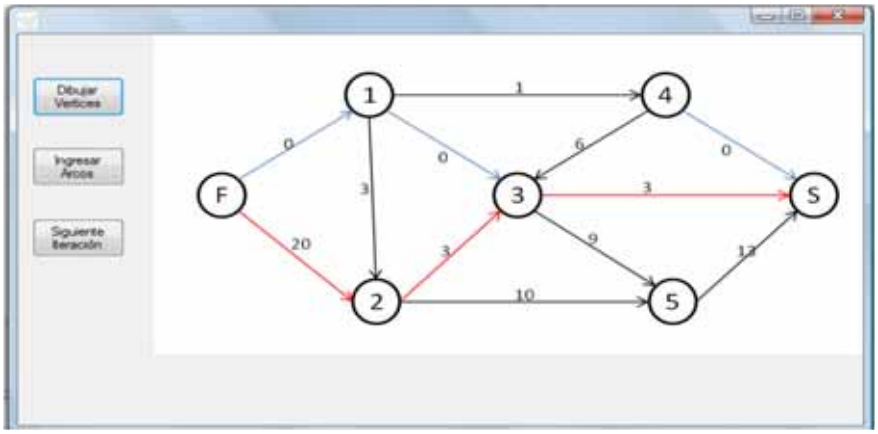


.

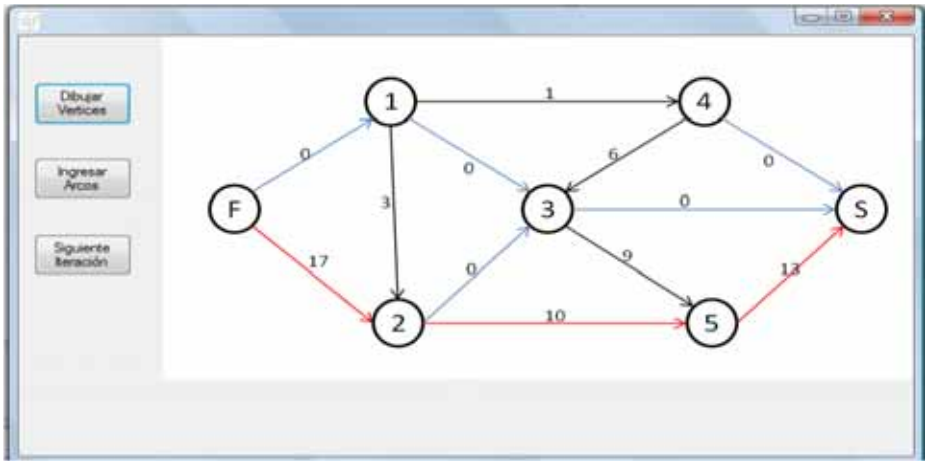
La nueva trayectoria es F-1-4-S, el flujo seleccionado es 7, se marca con azul el arco que ya no tiene capacidad para algún flujo, el acumulado es 12.



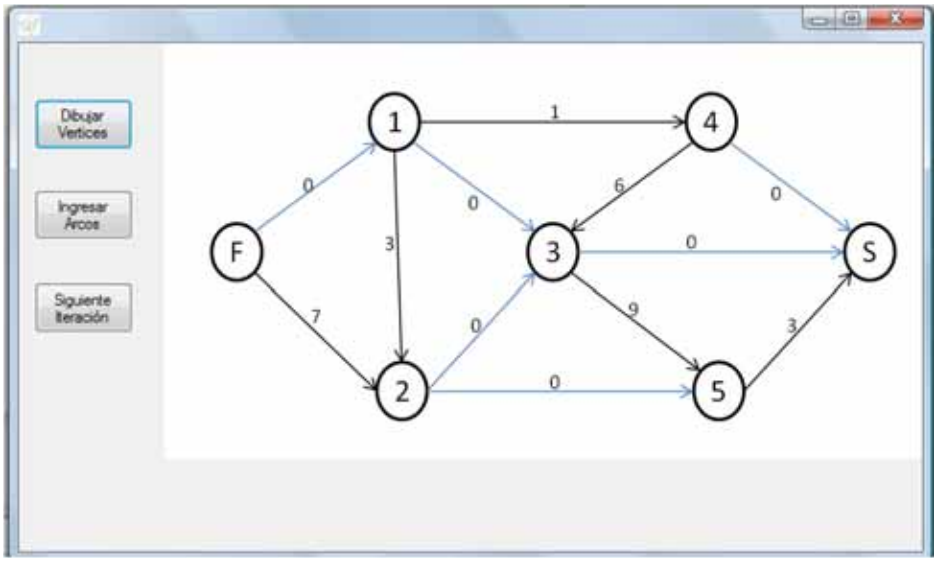
El nuevo camino que se encontró es F-2-3-S, nuevo flujo es 3, el flujo total es 15



El camino encontrado es F-2-5-S, el flujo es 10 y el acumulado 25.



Finalmente observamos que no hay trayectoria aumentante, por lo tanto el algoritmo termina y flujo máximo de F a S es de 25 unidades.



Instalación

Instalación.

Debido a que GPL fue desarrollada con la versión gratuita del *Visual Studio de Microsoft,* no se puede generar una versión empaquetada para que el usuario pueda instalarlo en su computadora, debido a esto se proporcionara todos los archivos y librerías necesarias para su ejecución.

Se tiene contemplado subir a un servidor todo lo necesario para que GPL pueda ser ejecutado en cualquier computadora, así como un manual en el cual se indicara el nombre de los archivos y las rutas en las que debe ser colocados, en dicho manual está incluido el de usuario para que el alumno se familiarice con GPL, también se proporcionara un correo electrónico para que se consulten dudas o dificultades que se puedan llegar a tener, incluso para que se informe de errores que tenga la aplicación con el fin de mejorarlo cada día. Se debe mencionar que se seguirá trabajando en esta aplicación para mejorarla y que pueda ser usada por más personas más allá de los alumnos de la Universidad Autónoma Metropolitana.

El recurso estará disponible para el trimestre 09O, para que los alumnos que cursen la UEA investigación de Operaciones I le den el uso para el cual fue desarrollado, esperando también comentarios y sugerencias.

Conclusiones

En el presente reporte, se mencionan los principales algoritmos que se implementaron, así como los módulos que conforman GPL. Es muy importante mencionar que este software tiene muchas limitaciones, pero mas allá de eso, lo importante es que se consiga lo que se busca, y es mostrar al usuario la evolución de la solución de su programa lineal a través de los pasos del algoritmo que haya sido implementado y que éste tenga una mejor comprensión del problema.

La razón de ser de este proyecto han sido las graficas, normalmente los desarrolladores de software para programación lineal, se quedaron en mostrar el método gráfico en R^2, pero ¿Por qué hasta R^2? Porque no salir de lo cotidiano, fue la principal causa del desarrollo de GPL, y considero que se ha logrado, como se mencionó anteriormente, con ciertas limitaciones, pero considerando el poder de cómputo necesario para hacer gráficos, lo desarrollado me ha dejado satisfecho.

Por otro lado, ¿Sólo el algoritmo simplex tiene solución gráfica?, cuando se planteó la posibilidad de este proyecto terminal con mi asesor, inmediatamente pensó en las posibilidades para el problema de transporte, el de asignación saldría de la idea del transporte, el problema de flujo fue hecho retomando la idea de un paquete “Optimización en Redes” mencionado en la bibliografía.

Quizá las gráficas en la programación lineal no llamen la atención de muchos, pero a consideración propia son parte fundamental para la mejor comprensión de los problemas incluidos en GPL.

Bibliografía

- “Investigación de Operaciones”, Taha, Hamdy A. Editorial Pearson Educación, Séptima Edición, 2004.

- “Introducción a la Investigación de Operaciones”, Hillier, F. S. y G. J. Liberman, Mc Graw Hill, Octava Edición, México, 2006.

- Investigación de Operaciones”, Wayne L. Winston, Cuarta Edición, Grupo Editorial Iberoamérica, México, 2005.

- López Bracho, R., Ortuño Sánchez M. P., Carrillo Romero, F. y M.T. Rodríguez Martínez, Paquete Computacional: Optimización en Redes (Versión 2) para Windows, UAM-A, México, 1997.

- www.zedgraph.org/

- http://csclab.murraystate.edu/bob.pilgrim/445/munkres.html

- http://operativa.tripod.com/invop/Invop.html