



Informe

Proyecto Final

Problema Morrales

Presentado por:

María Alejandra Pabón 1310263

Mayerly Suarez 1310284

Complejidad y Optimización

Irene Tischer

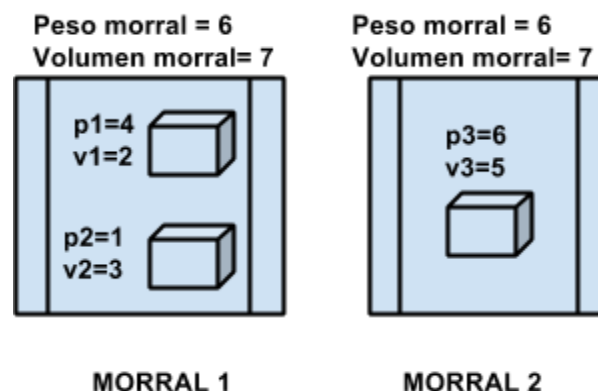
Informe Proyecto Final

Problema Morrales

1. DESCRIPCIÓN DEL PROBLEMA

Para la atención de un desastre se necesita calcular la cantidad óptima de personas que se requieren para transportar implementos de primeros auxilios agrupados en cajas hacia el sitio donde ocurrió una tragedia. Un equipo de rescate tiene que transportar a pie una serie de cajas con ítems de primeros auxilios a un sitio donde ocurrió un desastre. Cada caja (i) tiene un peso determinado (p_i) y un volumen determinado (v_i). Cada persona del equipo carga un morral con algunas cajas de primeros auxilios. La cantidad de cajas en cada morral está limitado por el volumen total que cabe en el morral (v) y por el peso total (p) que una persona puede cargar (v y p son iguales para todos los morrales).

Un ejemplo de distribución correcta de cajas dentro de cada morral se muestra a continuación:



El problema se divide principalmente en dos partes. A continuación se explica la descripción de cada parte con su modelo y más adelante unas pruebas y análisis correspondientes a cada una. También se nombran los detalles importantes de implementación relacionadas con las dos partes y en general de todo el proyecto.

De forma resumida, la primera parte de este problema consiste en resolver el problema de cantidad óptima de morrales para llevar las cajas, en otras palabras la cantidad óptima de personas que llevan las cajas. La segunda parte del problema consiste en encontrar una distribución equilibrada según los pesos de las cajas en los morrales.

2. MODELOS

2.1 DESCRIPCIÓN MODELO 1

El modelo de optimización consiste en minimizar la cantidad de personas que van a transportar los morrales, lo que es equivalente a reducir la cantidad de morrales que se necesitan para transportar todas las cajas. El modelo que se presenta a continuación es adecuado porque genera soluciones factibles y muestra la solución óptima al problema inicial.

A continuación se muestran las constantes, variables de decisión, función objetivos y restricciones del modelo de optimización que se propone

2.2 CONSTANTES MODELO 1

Tabla 1. Constantes Modelo 1

Nombre	Descripción
P	Peso máximo de cada uno de los morrales. $P > 0^*$
V	Volumen máximo de cada uno de los morrales. $V > 0^*$
p_i	Peso de la caja i . $p_i > 0$; $1 \leq i \leq n$
v_i	Volumen de la caja i . $v_i > 0$; $1 \leq i \leq n$
n	Cantidad de cajas en total. $n > 0$
M	Variable lo suficientemente grande

*El peso y el volumen máximo de cada morral es igual para todos los morrales.

:

2.3. VARIABLES DE DECISIÓN MODELO 1

Tabla 2. Variables de decisión Modelo 1

Nombre	Descripción
X_{ij} $0 \leq i \leq n$ $0 \leq j \leq n$	Variable binaria que dice si la caja i está en el morral j. Tiene valor 1 si la caja i está en el morral j Tiene valor 0 si la caja i no está en el morral j
Y_j $0 \leq j \leq n$	Variable binaria que dice si el morral j está vacío o no. Tiene valor 1 si el morral j no está vacío Tiene valor 0 si el morral j está vacío

2.4 FUNCIÓN OBJETIVO MODELO 1

$$\min z = \sum_j^n Y_j$$

La función objetivo corresponde a minimizar cantidad de morrales no vacíos, es decir a minimizar la sumatoria de morrales no vacíos (donde $Y_j=1$). Esto sería equivalente a decir que se minimizan la cantidad de personas que transportan los morrales, lo cual corresponde a lo que se pide inicialmente.

2.5 RESTRICCIONES MODELO 1

Tabla 3. Restricciones Modelo 1

N°	Restricción	Descripción
1	$\sum_j^n X_{ij} = 1$ $0 \leq X_{ij} \leq 1$	La restricción asegura que la caja i por lo menos se encuentre en un morral j; es decir; controla que no se repita esa caja en un morral y que se lleve la caja Esta restricción debe repetirse según el número de cajas.
2	$\sum_i^n X_{ij} p_i \leq P$ $0 \leq X_{ij} \leq 1$ $p_i \leq 0$ $P \leq 0$	Controla que las cajas con peso p_i no exceda la capacidad del morral j con peso P . Esta restricción debe repetirse según el número de cajas.
3	$\sum_i^n X_{ij} v_i \leq V$ $0 \leq X_{ij} \leq 1$ $v_i \leq 0$ $V \leq 0$	Controla que las cajas con volumen v_i no exceda la capacidad del morral j con volumen V . Esta restricción debe repetirse según el número de cajas.

4	$\sum_{ij}^n X_{ij} \leq M Y_j$	Esta restricción garantiza que el morral no está vacío. Cuando Y_j es 0 ; es decir, el morral j no contiene ninguna caja y hace un lado de la restricción 0; por lo tanto, la sumatoria de X_{ij} es también cero. De lo contrario, si el Y_j es 1, el morral contiene al menos una caja, entonces la sumatoria de cajas en el morral j no es 0, pero debe ser menor a un M (valor muy grande) multiplicado por $Y_j = 1$. Esta restricción debe repetirse según el número de cajas.
---	---------------------------------	---

2.3. DESCRIPCIÓN DEL MODELO 2

Luego de tener definidas la cantidad de personas que se requieren para atender el desastre, es necesario distribuir equilibradamente la carga contenida en cada morral minimizando la máxima diferencia entre los pesos totales, de tal manera que el peso del morral de cada persona que compone el equipo de rescate sea lo más equitativo posible.

A continuación se muestran las constantes, variables de decisión, función objetivos y restricciones del modelo que se propone:

2.4 CONSTANTES MODELO 2

Tabla 4. Constantes Modelo 2

Nombre	Descripción
P	Peso máximo de cada uno de los morrales. $P > 0^*$
V	Volumen máximo de cada uno de los morrales. $V > 0^*$
p_i	Peso de la caja i. $p_i > 0$; $1 \leq i \leq n$
v_i	Volumen de la caja i. $v_i > 0$; $1 \leq i \leq n$
n	Cantidad de cajas en total. $n > 0$
m	Cantidad total óptima de personas o morrales que se necesitan para transportar n cajas. Este valor es el resultado de la función objetivo del modelo 1.
$maxd$	Variable que dice la máxima diferencia entre las restas de cada uno de los morrales con cada uno de los demás morrales. Es la variable que se usa en la función objetivo. Inicialmente tiene valor de 1 y se controla con las restricciones que calculan las diferencias entre los morrales, esas diferencias son menores o iguales a esa máxima diferencia.

*El peso y el volumen máximo de cada morral es igual para todos los morrales.

Para el cálculo de abs se tuvo en cuenta la verdadera diferencia entre la distribución de los cajas que arroja el Modelo 1; es decir, con la carga total de cada morral se realizan las restas en valor absoluto obteniendo un valor constante, lo que finalmente contribuye a formar las restricciones que aseguran la construcción de una distribución de cargas más equilibrada, asegurando que las diferencias van a ser mejores a la distribución encontrada cuando se buscaba la cantidad óptima de morrales.

2.5 VARIABLES DE DECISIÓN MODELO 2

Tabla 5. Variables de decisión Modelo 2

Nombre	Descripción
X_{ij} $0 \leq i \leq n$ $0 \leq j \leq m$	Variable binaria que dice si la caja i está en el morral j. Tiene valor 1 si la caja i está en el morral j Tiene valor 0 si la caja i no está en el morral j

2.6 FUNCIÓN OBJETIVO MODELO 2

$$\min z = \max d$$

La función objetivo corresponde a minimizar la máxima diferencia entre los pesos de los morrales, es decir, a minimizar el máximo resultado entre las restas del peso de cada uno de los morrales con cada uno de los otros pesos de los morrales.

2.7 RESTRICCIONES MODELO 2

Tabla 6. Restricciones Modelo 2

N°	Restricción	Descripción
1	$\sum_i^n X_{ij}p_i - \sum_i^n X_{ik} \leq \max d$ $0 \leq X_{ij} \leq 1$ <p>Donde j es un morral y k es otro morral</p>	Controla que la resta entre el peso de las cajas que hay un morral con el peso de las cajas dentro del otro morral no sea mayor a la máxima diferencia entre los pesos de los morrales. Esta restricción debe repetirse según la permutación mP2. En cada restricción de este tipo se hace la resta entre el

		<p>peso del morral j menos el peso del morral morral k, esta resta también se hace en sentido contrario y cuenta como otra restricción.. Entonces por ejemplo al tener 3 morrales se hace la resta entre el peso del morral 1 con el morral 2, morral 2 menos morral 1, morral 1 menos morral 3, morral 3 menos morral 1, morral 2 menos morral 3 y morral 3 menos morral 2, teniendo un total de 6 restricciones=3P2.</p>
2	$\sum_j^m X_{ij} = 1$ $0 \leq X_{ij} \leq 1$	<p>La restricción asegura que la caja i por lo menos se encuentre en un morral j; es decir; controla que no se repita esa caja en un morral y que se lleve la caja Esta restricción debe repetirse según el número de cajas.</p>
3	$\sum_i^n X_{ij} p_i \leq P$ $0 \leq X_{ij} \leq 1$ $P_i \leq 0$ $P \leq 0$	<p>Controla que las cajas con peso p_i no exceda la capacidad del morral j con peso P. Esta restricción debe repetirse según el número de morrales m</p>
4	$\sum_i^n X_{ij} v_i \leq V$ $0 \leq X_{ij} \leq 1$ $V_i \leq 0$ $V \leq 0$	<p>Controla que las cajas con volumen v_i no exceda la capacidad del morral j con volumen V. Esta restricción debe repetirse según el número de morrales m.</p>

3. DETALLES DE IMPLEMENTACIÓN

Como como lenguaje de programación para la implementación se escogió Java, el cual es compatible con la librería LpSolve como apoyo a la solución del modelo, esta librería incluye los algoritmos necesarios como Simplex, Simplex Dual y Branch and Bound para resolver el problema según la modelación de solución que se le envía. Las clases implementadas fueron: Modelo.java, Modelo2.java, Morral.java. De forma general, Modelo.java es la encargada de la creación del modelo 1 (restricciones, función objetivo, modelar, nombres de variables y clasificación de las cajas en los morrales). Modelo2.java es la encargada de la creación del modelo 2 (restricciones, función objetivo, modelar, nombres de variables y clasificación de las cajas en los morrales). En Morral.java se gestiona la interfaz gráficas con los datos y la carga del archivo de texto.

En la implementación primero se definieron las variables globales que consideramos necesarias para el almacenamiento de los datos cargados con el archivo de texto, modelación e impresión de datos y solución.

Para especificar en la construcción del modelo 1 (que procesa LPSolve), que se trabaja con variables binarias X_{ij} y Y_j se creó el método `declararBinarias()`. Para especificar en la construcción del modelo 2 (que procesa LPSolve), que se trabaja con variables binarias X_{ij} y la variable enteras $maxd$ se creó el método `declararVariables()`. Se usó también la variable global `resultadoTiempoEjecucion` tanto en el modelo 1 como en el modelo 2 para calcular el tiempo de ejecución del modelo con la librería `lp_solve` para Java.

Los valores de las constantes que se necesitan para la creación del modelo, las cuales se describieron en la tabla 1 del presente informe, se cargan por medio de un archivo `.txt` con el siguiente estándar dentro del archivo respecto a la forma en cómo se escriben los datos de las constantes:

```
numeroCajas
volumenMaximoMorrall
pesoMaximoMorrall
NumeroCaja VolumenCaja PesoCaja
NumeroCaja VolumenCaja PesoCaja
...
```

Para la traducción del modelo creado previamente en LPSolve IDE a `lp_solve` en Java, primero se importó la librería de `lp_solve` para Java y se agregaron dos archivos `.dll` en la ruta : `lpsolve55.dll` y `lpsolve55j.dll` en el sistema operativo Windows.

Se trabajo en la clase `Modelo.java` y se usaron los métodos `nombrarVariables()`, `declararBinarias()`, `armarFuncionObjetivo()`, `armarPrimeraRestriccion()`, `armaSegundaRestriccion()`, `armarTerceraRestriccion()`, `armarCuartaRestriccion()`, los cuales se llamaron en el método `modelar()`. Dentro de estos métodos se usaron las funciones correspondientes a la creación de restricciones y función objetivo, solución del modelo, impresión de la función objetivo y valores finales de la variables, mejora del Branch and Bound, entre otras que se consideraron usar para observar la construcción del modelo y su solución de la librería `lp_solve` de Java. También se usó la función que permite crear un archivo `modeloMorrall.lp` correspondiente al modelo que se construye, éste se crea dentro de la carpeta del proyecto `/src/morrall`, en este archivo se puede verificar que restricciones y función objetivo se crearon. Adicionalmente se creó el método `clasificarCajas(variablesdelmodelocreadas, resuladofuncionobjetivo)` para determinar la distribución de cajas en los morrales y así poder mostrar que cajas deben en ir finalmente en cada morral.

Se trabajo tambien en la clase `Modelo2.java` y se usaron los métodos `nombrarVariables()`, `declararVariables()`, `armarFuncionObjetivo()`, `armarRestriccionDistribucion()`, `armarPrimeraRestriccion()`, `armaSegundaRestriccion()`, `armarTerceraRestriccion()`, los cuales se llamaron en el método `modelar()`. Dentro de estos métodos se usaron las funciones correspondientes a la creación de restricciones y función objetivo, solución del modelo, impresión de la función objetivo y valores finales de la variables, mejora del

Branch and Bound, entre otras que se consideraron usar para observar la construcción del modelo y su solución de la librería lp_solve de Java. También se usó la función que permite crear un archivo modelo2Morrall.lp correspondiente al modelo que se construye, éste se crea dentro de la carpeta del proyecto /src/morrall, en este archivo se puede verificar que restricciones y función objetivo se crearon. También se creó el método clasificarCajas(variablesdelmodelocreadas) para determinar la distribución de cajas en los morrales y así poder mostrar que cajas deben ir finalmente en cada morral. En esta clase Modelo2.java se destaca que en los diferentes métodos tanto de la construcción de restricciones, como de función objetivo y clasificación de cajas en morrales fue importante la variable m, tomada del resultado óptimo de morrales de Modelo.java.

Para el modelo 1, respecto al cálculo de la M (grande) que se escogió se puede decir lo siguiente: la M es una variable que ayuda a determinar si el morral no está vacío, como se explica en la Restricción 4 (Ver tabla de restricciones del modelo 1). Por lo tanto, el valor de esta constante debe ser un número lo suficientemente grande para que la restricción se cumpla y el modelo de la solución sea consistente y como la función objetivo está minimizando por esto funciona también. Tomar un valor arbitrario, como 100000, 1000000, no es completamente factible para los casos en que se ingresan una cantidad de cajas cercanos a esos valores, si el programa soporta esta cantidad de entradas. pensando en el peor caso. Debido a esto se definió un valor para M que independientemente de la entrada asegura que la restricción, por ejemplo: la suma de los pesos y volúmenes de las cajas $M = \sum_i^n v_i + \sum_i^n p_i$. Incluso se cumple en el caso que los volúmenes y pesos ingresados sean 0, el modelo determina que la solución no es factible. Para realizar esto se creó el método definirM(pesosCajas, volúmenesCajas) en la clase Modelo.java.

Respecto a la interfaz se pide primero que se cargue un archivo con los datos de las constantes que sirven de entrada al modelo, al cargar en estos datos se muestran en dos tablas correspondientes a los datos de cada morral y de cada caja. Luego con el botón “calcular” se procesa el modelo implementado según los datos de entrada cargados y finalmente en otra tabla muestra la distribución de las cajas en cada morral y el resultado de la solución óptima correspondiente a la cantidad mínima de morrales que se deben llevar según la cantidad de cajas en total. Se propone la opción “Limpiar tabla”, para cuando se desee cargar otro archivo con otros datos de entrada y ejecutar un nuevo modelo.

4. PRUEBAS

4.1 PRUEBAS PARTE 1

Prueba 1. Prueba de solución factible, pesos y volúmenes de las cajas pequeños en comparación al peso y volumen del morral

The screenshot shows a software window titled "Optimización Morrales" with a pink border. Inside, there's a header "Problema Morrales" with a small icon of a backpack. Below it is a file path input field containing "C:\\Users\\Maleja\\Documents\\p1_pvpequenos.txt" and an "Examinar" button. The main area is divided into two sections: "Datos Cajas" and "Datos Morral".

Datos Cajas

Nº	Peso	Volumen
1	2	1
2	1	1
3	2	2
4	1	3

Datos Morral

Numero Cajas	Peso	Volumen
4.0	5	10

Below these tables are three buttons: "Calcular", "Graficar", and "Limpiar datos".

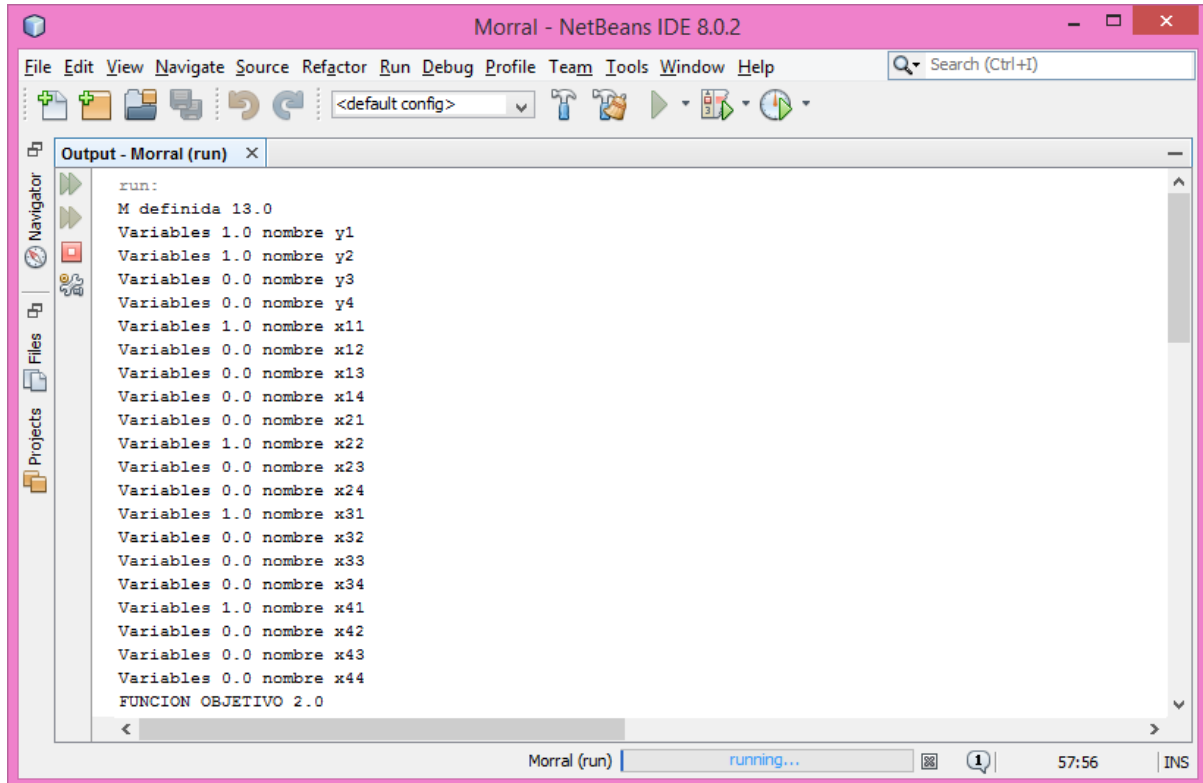
Distribución de las cajas en los morrales

Morral	Cajas
1	Caja 1;Caja 3;Caja 4;
2	Caja 2;

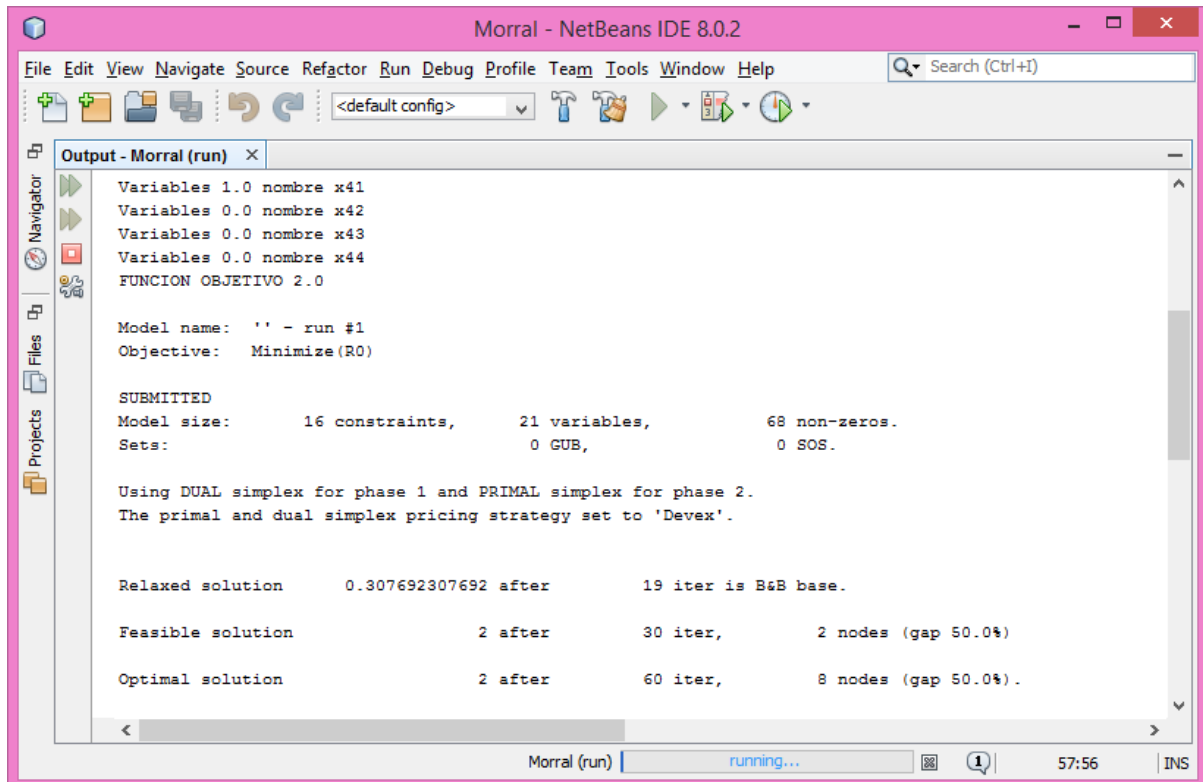
At the bottom, there are two input fields with their values: "Total minimo de personas" with value "2.0" and "(Total minimo de morrales)", and "Tiempo de ejecucion" with value "0.059" and "(ms)".

Es una prueba básica de un caso promedio que demuestra que el programa y el modelo matemático se comportan correctamente al encontrar una solución óptima factible transportando 4 cajas en 2 morrales, donde los pesos y volúmenes de cada caja no exceden la capacidad máxima del morral. Además, puede evidenciarse en la distribución de las cajas dentro de cada morral que la sumatoria de pesos y volúmenes de las cajas contenidas no excede la capacidad del morral. La solución es encontrada en un tiempo razonable de 0,059 teniendo en cuenta el tamaño de las entradas. Cabe resaltar que es una prueba con valores pequeños para los volúmenes y pesos de cajas; es decir, que están lejos de exceder el límite de la capacidad del morral.

A continuación se muestra la evidencia de los valores que arroja la librería LP Solve para Java, luego de procesar el archivo .lp que se crea con los datos ingresados.



```
run:
M definida 13.0
Variables 1.0 nombre y1
Variables 1.0 nombre y2
Variables 0.0 nombre y3
Variables 0.0 nombre y4
Variables 1.0 nombre x11
Variables 0.0 nombre x12
Variables 0.0 nombre x13
Variables 0.0 nombre x14
Variables 0.0 nombre x21
Variables 1.0 nombre x22
Variables 0.0 nombre x23
Variables 0.0 nombre x24
Variables 1.0 nombre x31
Variables 0.0 nombre x32
Variables 0.0 nombre x33
Variables 0.0 nombre x34
Variables 1.0 nombre x41
Variables 0.0 nombre x42
Variables 0.0 nombre x43
Variables 0.0 nombre x44
FUNCION OBJETIVO 2.0
```



```
Variables 1.0 nombre x41
Variables 0.0 nombre x42
Variables 0.0 nombre x43
Variables 0.0 nombre x44
FUNCION OBJETIVO 2.0

Model name: '' - run #1
Objective: Minimize (R0)

SUBMITTED
Model size:      16 constraints,      21 variables,      68 non-zeros.
Sets:           0 GUB,              0 SOS.

Using DUAL simplex for phase 1 and PRIMAL simplex for phase 2.
The primal and dual simplex pricing strategy set to 'Devex'.

Relaxed solution      0.307692307692 after      19 iter is B&B base.

Feasible solution      2 after      30 iter,      2 nodes (gap 50.0%)

Optimal solution      2 after      60 iter,      8 nodes (gap 50.0%).
```

Prueba 2. Prueba de solución factible, pesos y volúmenes de las cajas grandes en comparación al peso y volumen del morral

Optimizacion Morrales

Problema Morrales

C:\Users\Maleja\Documents\p2_pvgrandes.txt

N°	Peso	Volumen
1	2	20
2	3	4
3	25	5
4	10	1
5	13	5
6	4	10
7	8	5
8	13	1
9	12	10

Numero Cajas	Peso	Volumen
10.0	25	20

Distribución de las cajas en los morrales

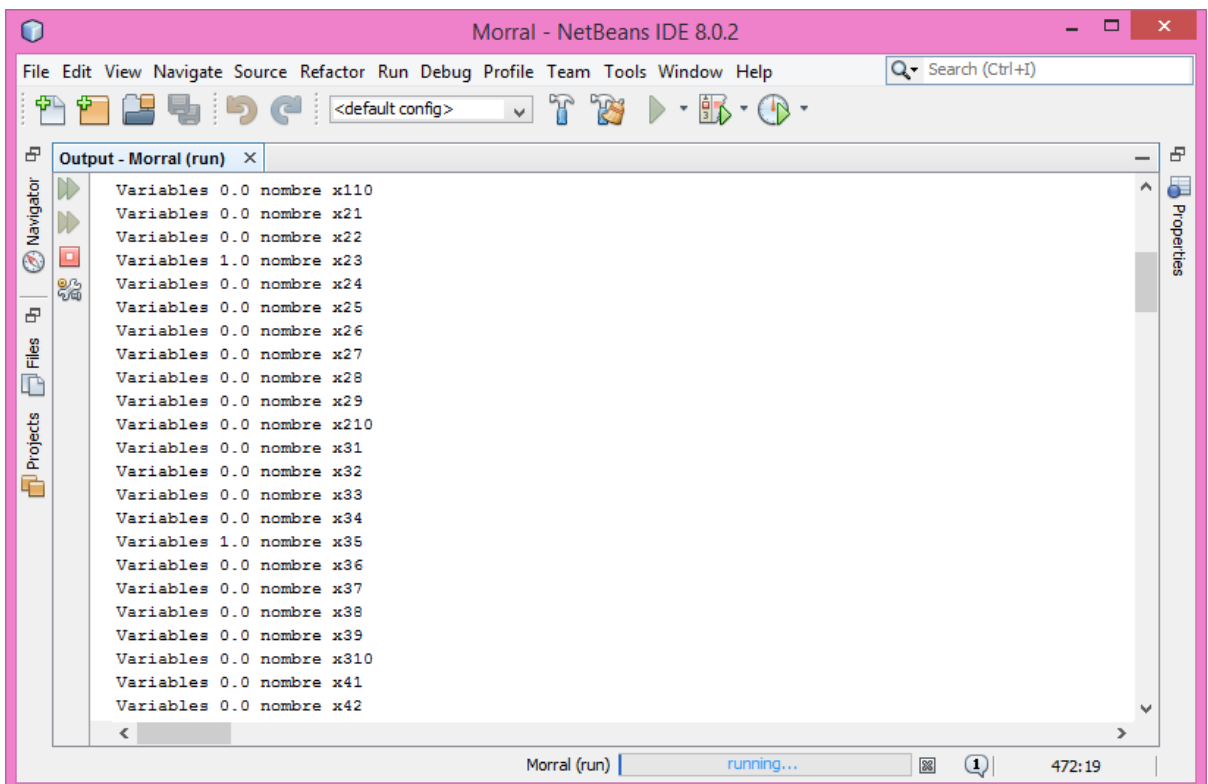
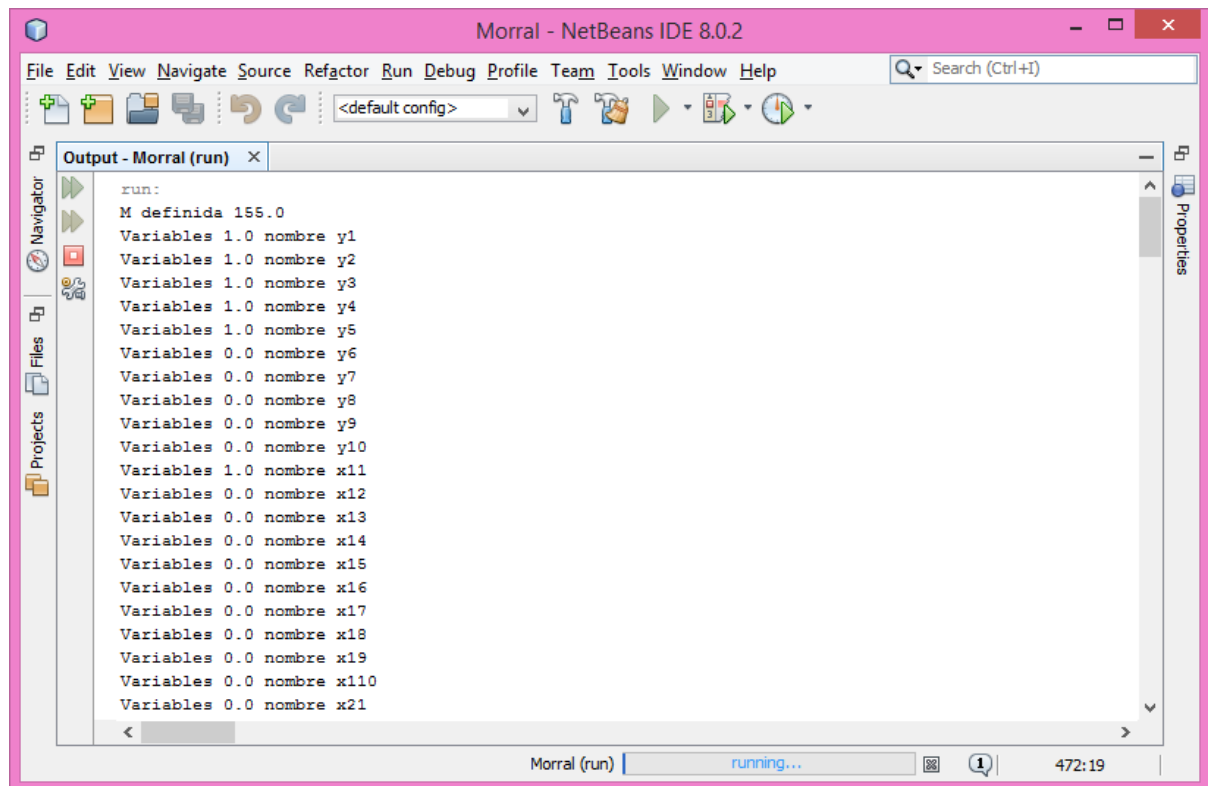
Morral	Cajas
1	Caja 1;
2	Caja 5;Caja 9;
3	Caja 2;Caja 6;Caja 7;Caja 10;
4	Caja 4;Caja 8;
5	Caja 3;

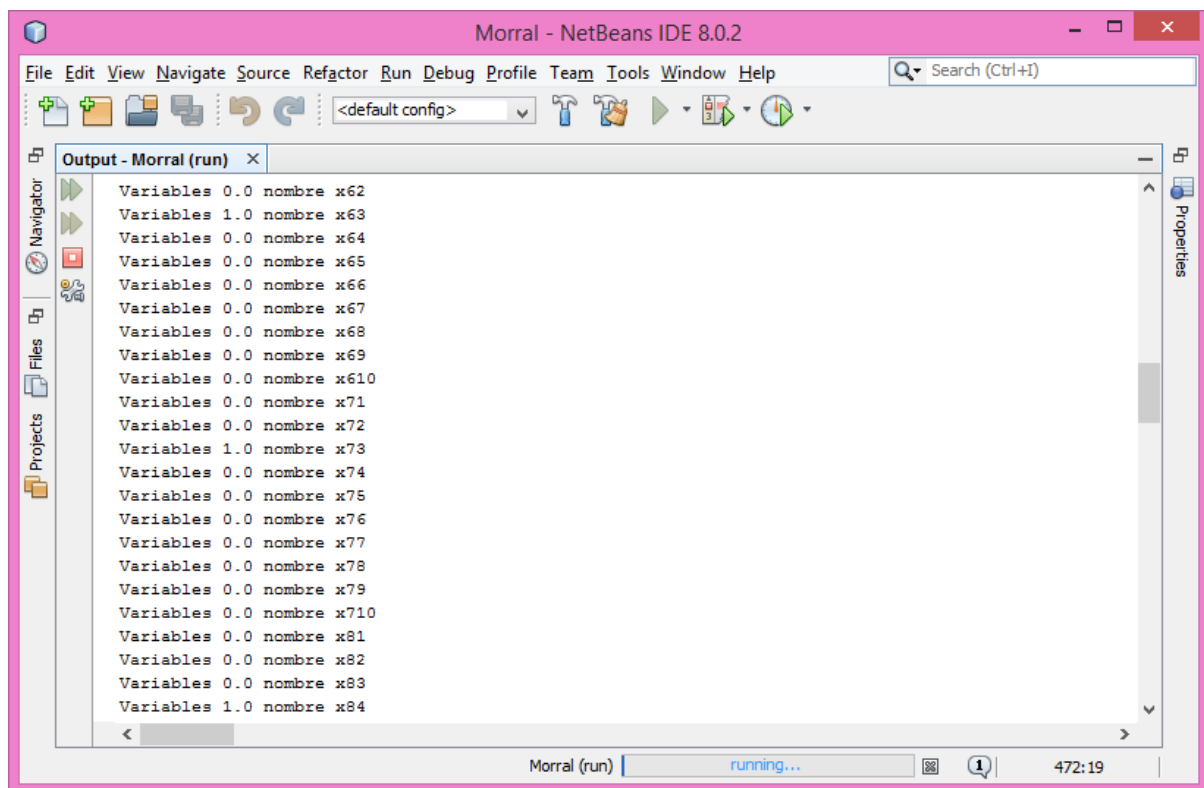
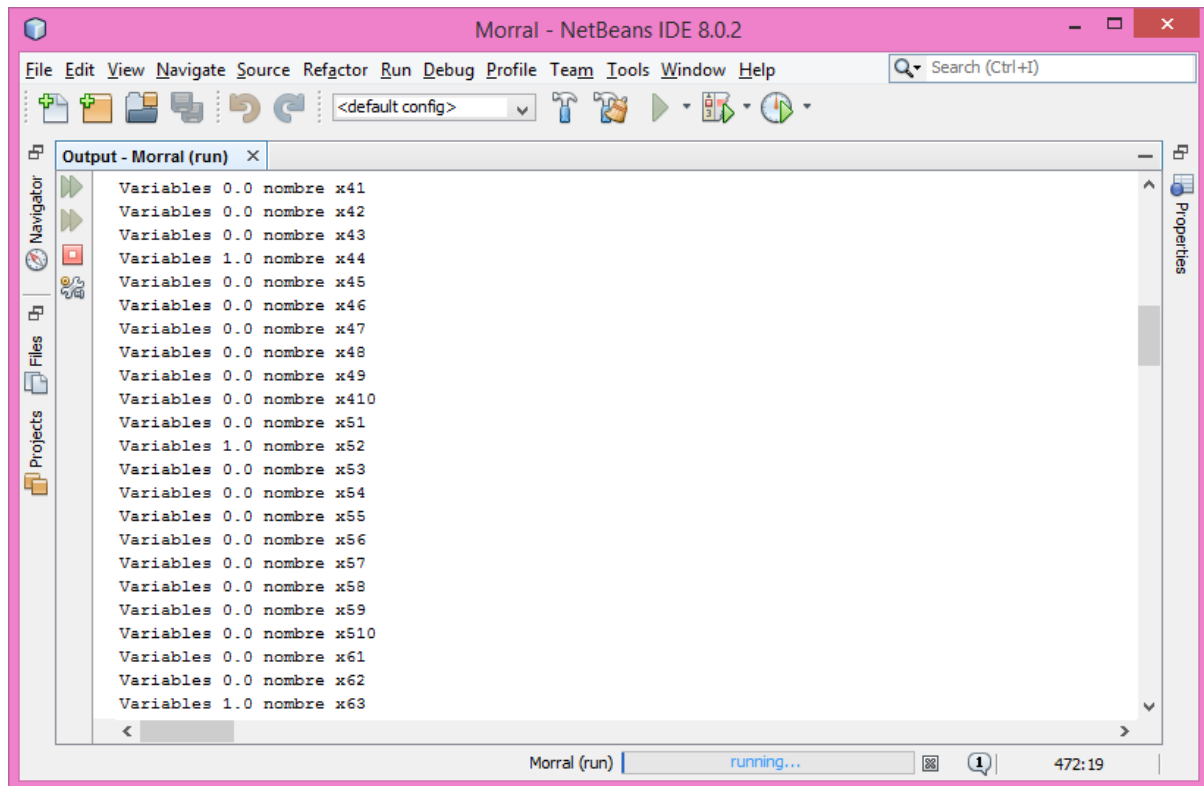
Total minimo de personas 5.0 (Total minimo de morrales)

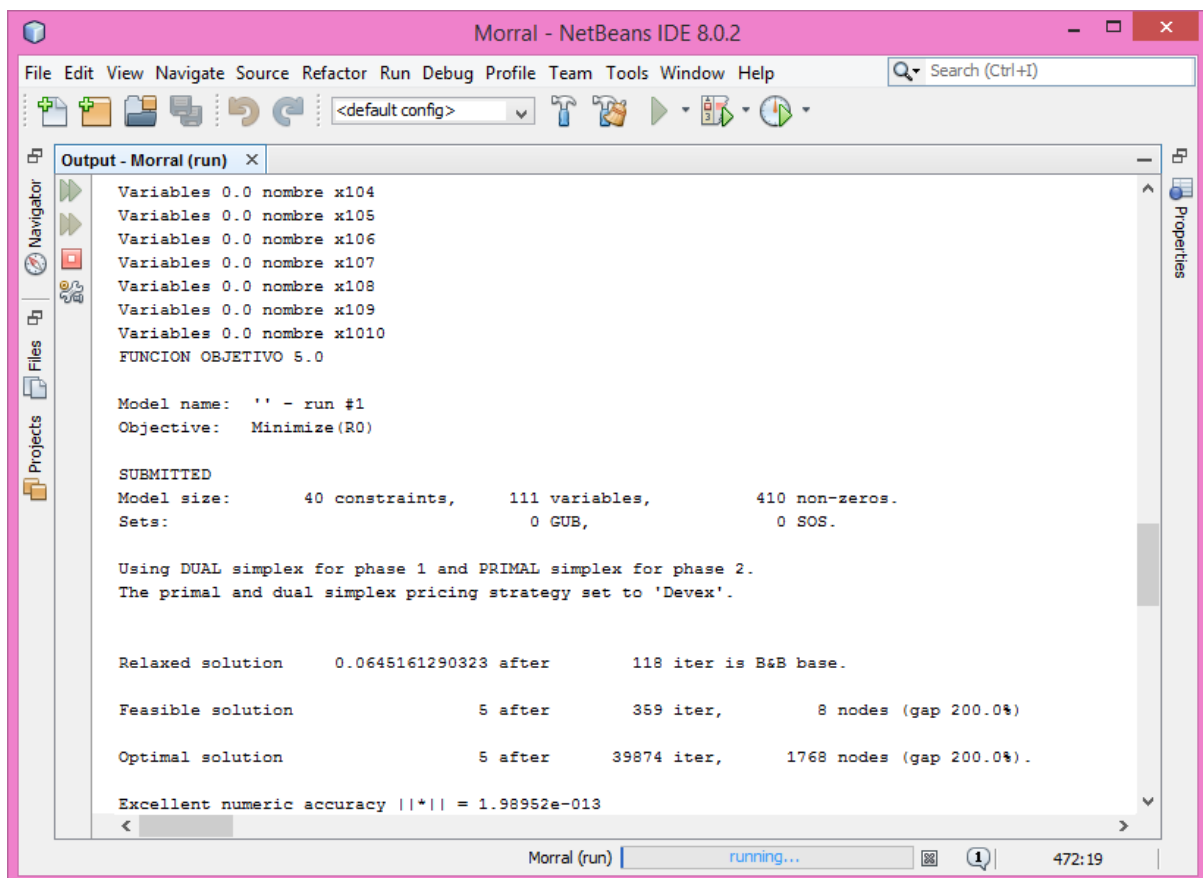
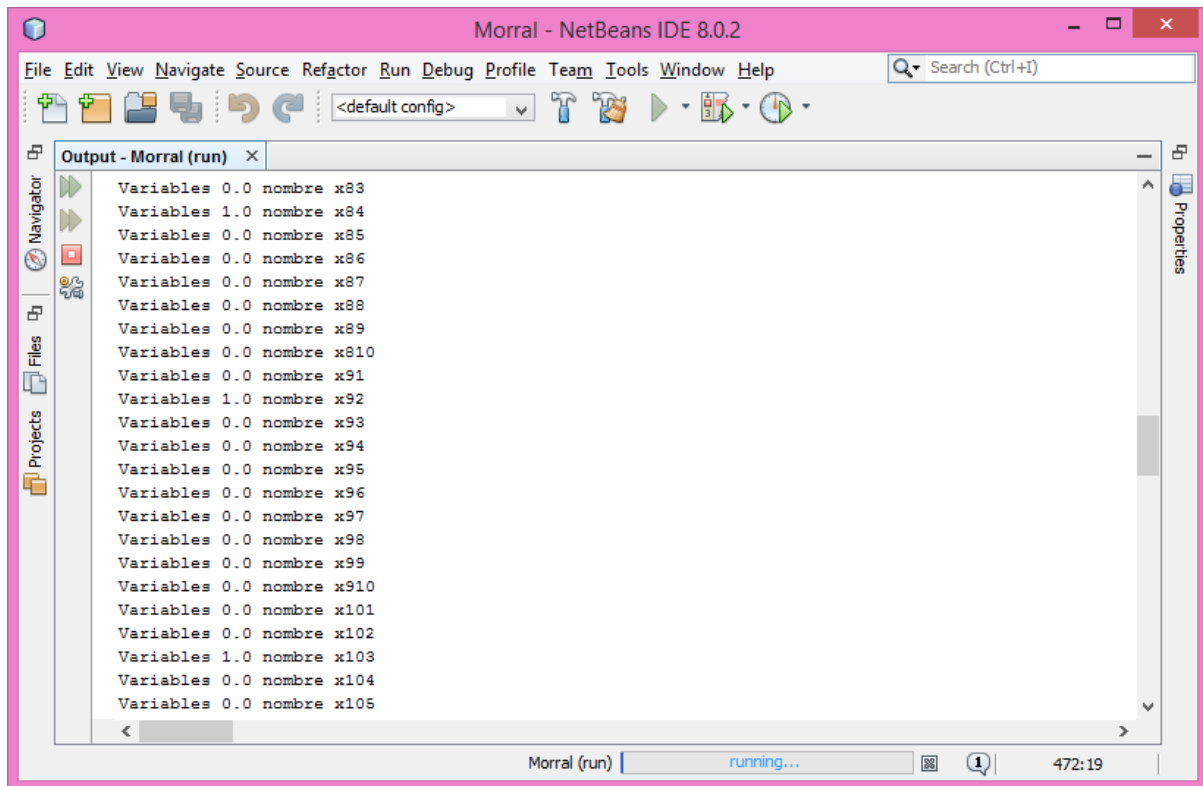
Tiempo de ejecucion 1.291 (ms)

Es la prueba de un caso promedio con algunos valores grandes para los volúmenes que se acercan al límite de la capacidad del morral, donde se muestra que el programa y el modelo matemático se comportan correctamente al encontrar una solución óptima factible con un número mayor de cajas, 10 cajas, que son transportadas en 5 morrales de capacidad considerablemente mayor con respecto a la prueba anterior. Los pesos y volúmenes de cada caja no exceden la capacidad máxima del morral. Además, puede evidenciarse en la distribución de las cajas dentro de cada morral que la sumatoria de pesos y volúmenes de las cajas contenidas no excede la capacidad del morral. La solución es encontrada en un tiempo razonable de 1,291 teniendo en cuenta que aumenta el tamaño de las entradas.

A continuación se muestra la evidencia de los valores arrojados por consola la librería LP Solve para Java luego de procesar el archivo .lp que se crea con los datos ingresados. La información obtenida además del valor de la Función Objetivo y el valor de M, es útil para corroborar los morrales no vacíos y la distribución de las cajas teniendo en cuenta el orden de los índices de la variable X_{ij}







Prueba 3. Prueba de solución factible , pesos y volúmenes de las cajas cercanos al valor del peso y volumen del morral

Es una prueba del peor caso con valores grandes para los volúmenes y pesos que se acercan al límite de la capacidad del morral, donde se muestra que el programa y el modelo matemático se comportan correctamente al encontrar una solución óptima factible con 6 cajas, que son transportadas en 6 morrales. Los pesos y volúmenes de cada caja no exceden la capacidad máxima del morral. Además, puede evidenciarse en la distribución de las cajas dentro de cada morral que la sumatoria de pesos y volúmenes de las cajas contenidas no excede la capacidad del morral. La solución es encontrada en un tiempo razonable de 0,134 teniendo en cuenta que el tamaño de las entradas es menor comparado con el caso anterior se puede explicar que el tiempo de ejecución es menor, aunque es un ejemplo del peor caso.

Optimizacion Morrales

Problema Morrales

C:\Users\Maleja\Documents\p3_pvcercanosaPV.txt

Datos Cajas		
Nº	Peso	Volumen
1	6	14
2	5	13
3	6	14
4	5	13
5	6	14
6	5	13

Datos Morral		
Numero Cajas	Peso	Volumen
6.0	7	15

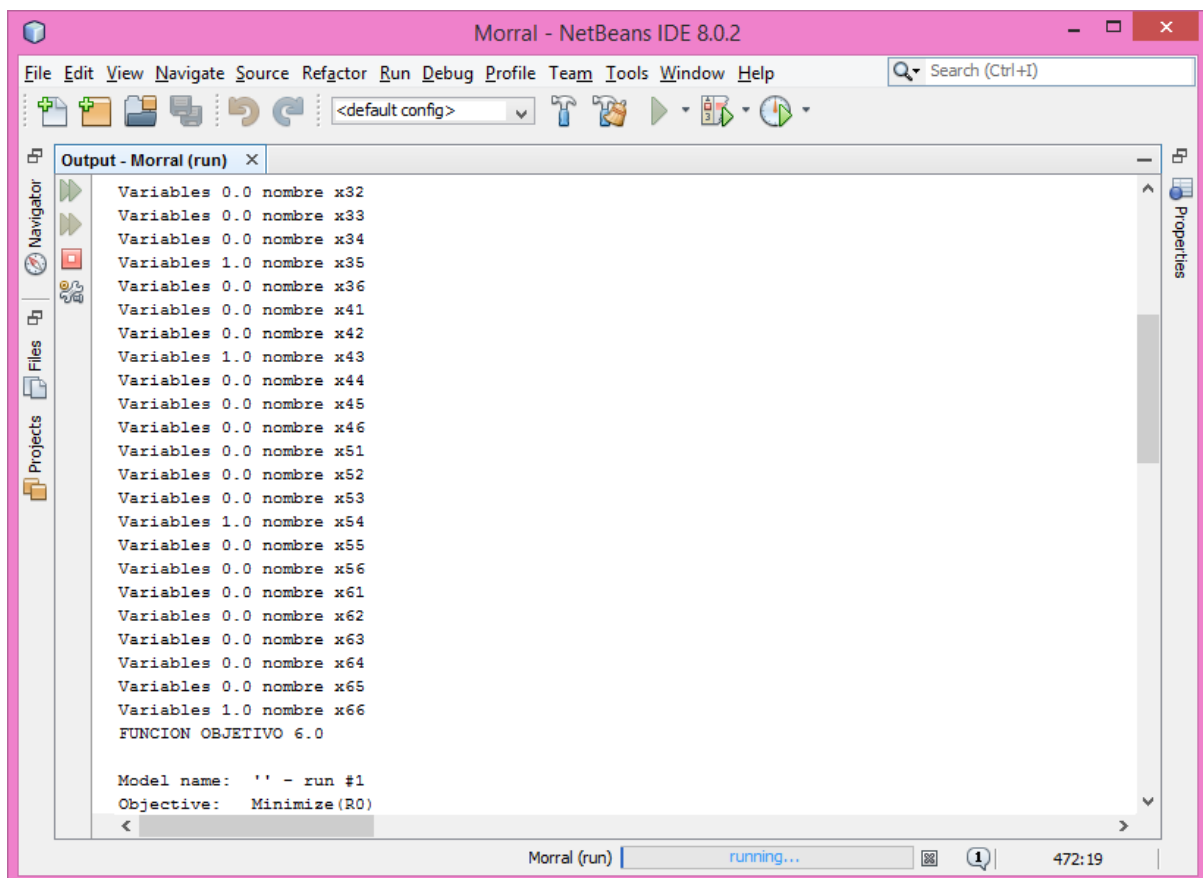
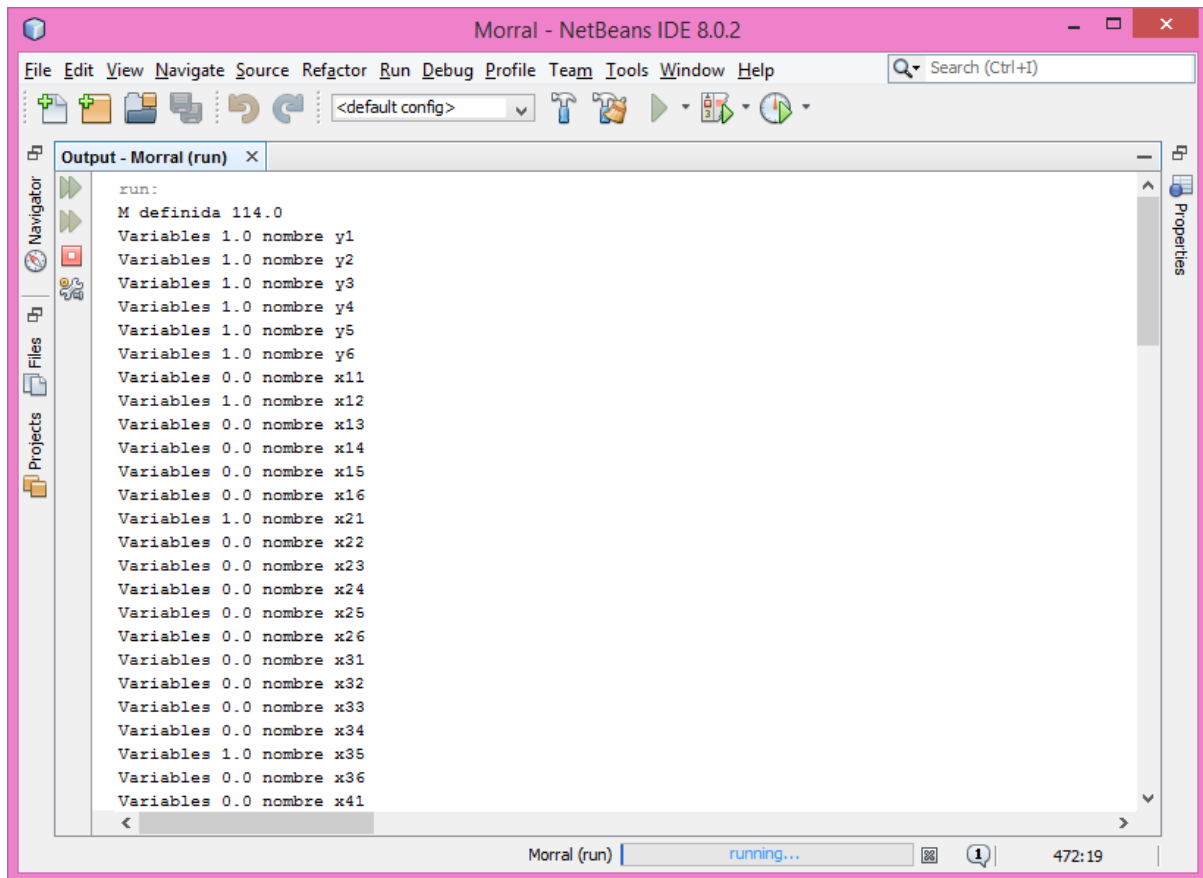
Distribución de las cajas en los morrales

Morral	Cajas
1	Caja 2;
2	Caja 1;
3	Caja 4;
4	Caja 5;
5	Caja 3;
6	Caja 6;

Total minimo de personas (Total minimo de morrales)

Tiempo de ejecucion (ms)

A continuación se muestra la evidencia la salida por consola de los valores arrojados la librería LP Solve y la forma como se construye el modelo del archivo .lp donde está contenida la función objetivo y las restricciones que procesa la librería anteriormente mencionada.



Morral - NetBeans IDE 8.0.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+F)

Modelo.java Morral.java modeloMorral.lp Output - Morral (run)

Source History

```
1  /* Objective function */
2  min: +y1 +y2 +y3;
3
4  /* Constraints */
5  +x11 +x12 +x13 = 1;
6  +x21 +x22 +x23 = 1;
7  +x31 +x32 +x33 = 1;
8  +6 x11 +4 x21 +4 x31 <= 6;
9  +6 x12 +4 x22 +4 x32 <= 6;
10 +6 x13 +4 x23 +4 x33 <= 6;
11 +15 x11 +x21 +x31 <= 15;
12 +15 x12 +x22 +x32 <= 15;
13 +15 x13 +x23 +x33 <= 15;
14 -31 y1 +x11 +x21 +x31 <= 0;
15 -31 y2 +x12 +x22 +x32 <= 0;
16 -31 y3 +x13 +x23 +x33 <= 0;
17
18 /* Variable bounds */
19 y1 <= 1;
20 y2 <= 1;
21 y3 <= 1;
22 x11 <= 1;
23 x12 <= 1;
24 x13 <= 1;
25 x21 <= 1;
26 x22 <= 1;
27 x23 <= 1;
28 x31 <= 1;
29 x32 <= 1;
30 x33 <= 1;
```

Find: variables Previous Next No matches

Morral (run) running... 25:10 INS

Morral - NetBeans IDE 8.0.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+F)

Modelo.java Morral.java modeloMorral.lp Output - Morral (run)

Source History

```
13 +15 x13 +x23 +x33 <= 15;
14 -31 y1 +x11 +x21 +x31 <= 0;
15 -31 y2 +x12 +x22 +x32 <= 0;
16 -31 y3 +x13 +x23 +x33 <= 0;
17
18 /* Variable bounds */
19 y1 <= 1;
20 y2 <= 1;
21 y3 <= 1;
22 x11 <= 1;
23 x12 <= 1;
24 x13 <= 1;
25 x21 <= 1;
26 x22 <= 1;
27 x23 <= 1;
28 x31 <= 1;
29 x32 <= 1;
30 x33 <= 1;
31
32 /* Integer definitions */
33 int y1,y2,y3,x11,x12,x13,x21,x22,x23,x31,x32,x33;
34
```

Find: variables Previous Next No matches

Morral (run) running... 25:10 INS

Prueba 4. Prueba de solución factible , pesos y volúmenes de las cajas iguales al valor del peso y volumen del morral

Es una prueba del peor caso, porque se debe emplear un morral para llevar cada caja. Se ingresa un valor de volumen de caja igual a la capacidad del volumen del morral, adicionalmente los pesos de las cajas se acercan al límite de la capacidad del morral en peso, razón por la cual cada caja debe ir en un morral. El programa y el modelo matemático se comportan correctamente al encontrar una solución óptima factible con 3 cajas, que son transportadas en 3 morrales. Los pesos y volúmenes de cada caja no exceden la capacidad máxima del morral. Además, puede evidenciarse en la distribución de las cajas dentro de cada morral que la sumatoria de pesos y volúmenes de las cajas contenidas no excede la capacidad del morral. La solución es encontrada en un tiempo razonable de 0,039 ms teniendo en cuenta que aunque es un ejemplo del peor caso, el tamaño de las entradas es menor comparado con el caso anterior.

Optimizacion Morrales

Problema Morrales

::\Users\Maleja\Documents\p4_pvigualesaPV - copia.txt **Examinar**

Datos Cajas

N°	Peso	Volumen
1	6	15
2	4	1
3	4	1

Datos Morral

Numero Cajas	Peso	Volumen
3.0	6	15

Calcular **Graficar** **Limpiar datos**

Distribución de las cajas en los morrales

Morral	Cajas
1	Caja 2;
2	Caja 1;
3	Caja 3;

Total minimo de personas 3.0 (Total minimo de morrales)

Tiempo de ejecucion 0.039 (ms)

Morral - NetBeans IDE 8.0.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

<default config>

Output - Morral (run)

```
Variables 1.0 nombre x12
Variables 0.0 nombre x13
Variables 1.0 nombre x21
Variables 0.0 nombre x22
Variables 0.0 nombre x23
Variables 0.0 nombre x31
Variables 0.0 nombre x32
Variables 1.0 nombre x33
FUNCION OBJETIVO 3.0

Model name: '' - run #1
Objective: Minimize(R0)

SUBMITTED
Model size:      12 constraints,      13 variables,      39 non-zeros.
Sets:           0 GUB,              0 SOS.

Using DUAL simplex for phase 1 and PRIMAL simplex for phase 2.
The primal and dual simplex pricing strategy set to 'Devex'.

Relaxed solution      0.0967741935484 after      11 iter is B&B base.

Feasible solution          3 after      27 iter,      7 nodes (gap 100.0%)

Optimal solution          3 after      35 iter,      12 nodes (gap 100.0%).
```

Morral (run) running... 472:19

Morral - NetBeans IDE 8.0.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

<default config>

Output - Morral (run)

```
run:
M definida 31.0
Variables 1.0 nombre y1
Variables 1.0 nombre y2
Variables 1.0 nombre y3
Variables 0.0 nombre x11
Variables 1.0 nombre x12
Variables 0.0 nombre x13
Variables 1.0 nombre x21
Variables 0.0 nombre x22
Variables 0.0 nombre x23
Variables 0.0 nombre x31
Variables 0.0 nombre x32
Variables 1.0 nombre x33
FUNCION OBJETIVO 3.0

Model name: '' - run #1
Objective: Minimize(R0)

SUBMITTED
Model size:      12 constraints,      13 variables,      39 non-zeros.
Sets:           0 GUB,              0 SOS.

Using DUAL simplex for phase 1 and PRIMAL simplex for phase 2.
The primal and dual simplex pricing strategy set to 'Devex'.
```

Morral (run) running... 472:19

Prueba 5. Prueba de solución no factible, alguno(s) de los pesos de las cajas es más grande que el peso máximo de un morral.

Es una prueba para evidenciar que el programa y el modelo matemático se comportan correctamente al no encontrar una solución factible para los parámetros ingresados, específicamente, el peso de las cajas 1 y 3 excede la capacidad máxima en peso del morral.

The screenshot shows the 'Optimizacion Morrales' application window. At the top, there's a title bar and a header area with a backpack icon and the title 'Problema Morrales'. Below this, a file path 'C:\Users\Maleja\Documents\p5_nofactiblemayorP.txt' is entered, with an 'Examinar' button. The main area is divided into two sections: 'Datos Cajas' and 'Datos Morral'. 'Datos Cajas' contains a table with 4 rows and 3 columns (Nº, Peso, Volumen). 'Datos Morral' contains a table with 3 columns (Numero Cajas, Peso, Volumen) and one row with values 4.0, 6, and 15. Below these tables are buttons for 'Calcular', 'Graficar', and 'Limpiar datos'. A 'Mensaje' dialog box is open in the center, displaying an information icon and the text 'La solucion no es factible', with an 'Aceptar' button. At the bottom, there are input fields for 'Total minimo de personas' and 'Tiempo de ejecucion', both with '(ms)' in parentheses.

Nº	Peso	Volumen
1	8	1
2	5	4
3	17	1
4	5	4

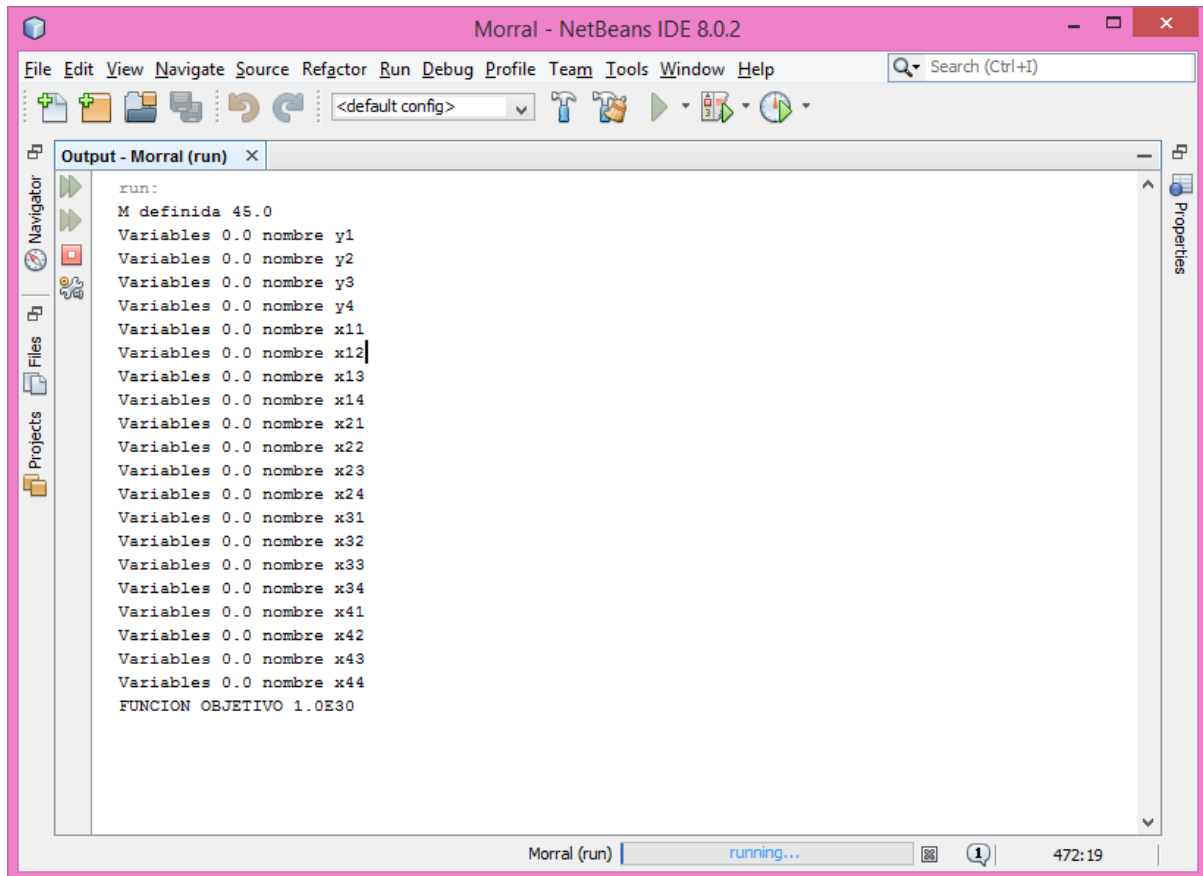
Numero Cajas	Peso	Volumen
4.0	6	15

Mensaje

La solucion no es factible

Aceptar

En consecuencia, el modelo se construye pero la salida por consola devuelve todas las variables binarias es 0; es decir, no se usa ningún morral.




```
run:
M definida 45.0
Variables 0.0 nombre y1
Variables 0.0 nombre y2
Variables 0.0 nombre y3
Variables 0.0 nombre y4
Variables 0.0 nombre x11
Variables 0.0 nombre x12
Variables 0.0 nombre x13
Variables 0.0 nombre x14
Variables 0.0 nombre x21
Variables 0.0 nombre x22
Variables 0.0 nombre x23
Variables 0.0 nombre x24
Variables 0.0 nombre x31
Variables 0.0 nombre x32
Variables 0.0 nombre x33
Variables 0.0 nombre x34
Variables 0.0 nombre x41
Variables 0.0 nombre x42
Variables 0.0 nombre x43
Variables 0.0 nombre x44
FUNCION OBJETIVO 1.0E30
```

The screenshot shows the NetBeans IDE 8.0.2 interface. The 'Output - Morral (run)' window is open, displaying the output of a Morral run. The output shows a list of variables and their values, all set to 0.0, indicating that no binary variables are used in the model. The variables listed are y1, y2, y3, y4, x11, x12, x13, x14, x21, x22, x23, x24, x31, x32, x33, x34, x41, x42, x43, and x44. The objective function value is 1.0E30. The status bar at the bottom indicates 'Morral (run)' is running.

Prueba 6. Prueba de solución no factible, alguno(s) de los volúmenes de las cajas es más grande que el volumen máximo de un morral.

Es una prueba para evidenciar que el programa y el modelo matemático se comportan correctamente al no encontrar una solución factible para los parámetros ingresados, específicamente, el volumen de las cajas 1 y 3 excede la capacidad máxima en volumen del morral.

Optimización Morrales

 **Problema Morrales**

C:\Users\Maleja\Documents\p6_nofactiblevmayorV.txt

Datos Cajas

Nº	Peso	Volumen
1	1	16
2	5	4
3	4	17
4	5	4

Datos Morral

Numero Cajas	Peso	Volumen
4.0	6	15


Distribución de las cajas

Morral

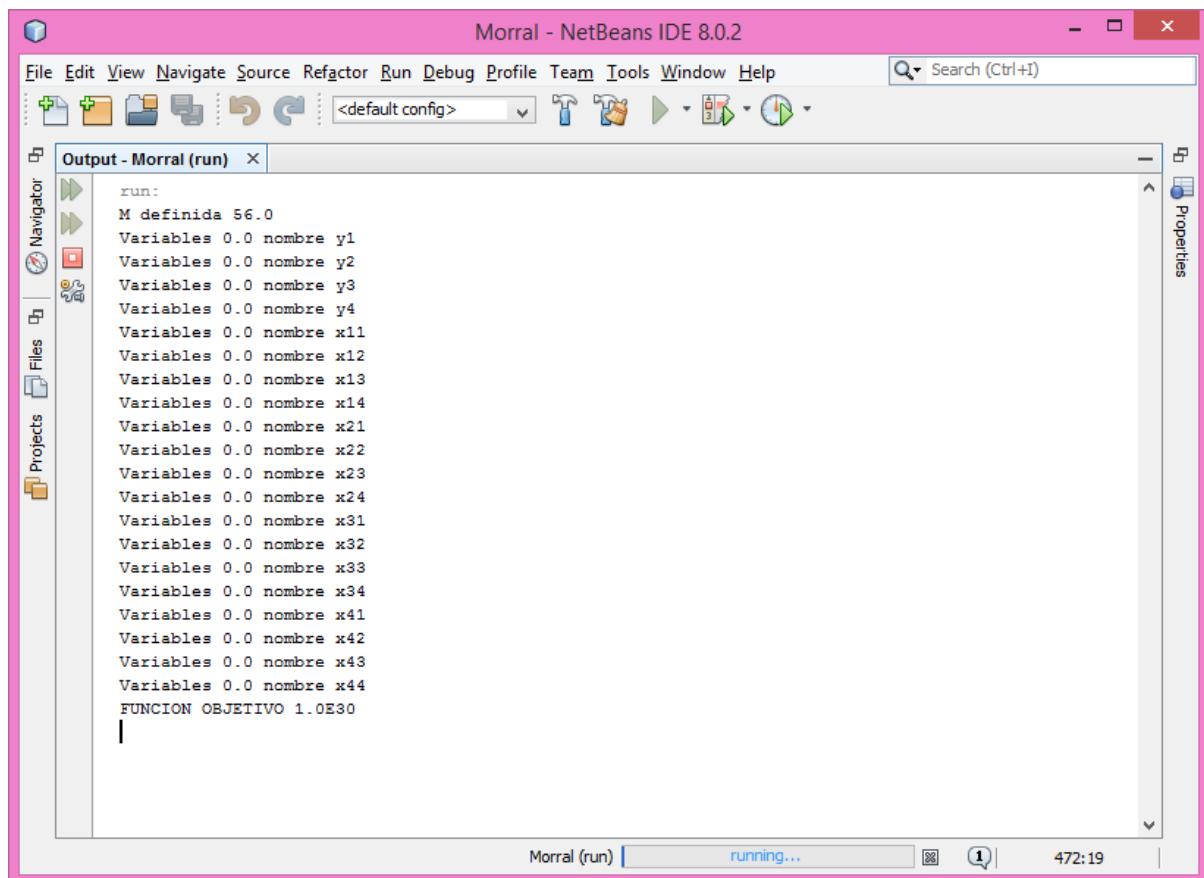
Total minimo de personas (Total minimo de morrales)

Tiempo de ejecucion (ms)

Mensaje

 La solucion no es factible

En consecuencia, el modelo se construye pero la salida por consola devuelve todas las variables binarias es 0; es decir, no se usa ningún morral.

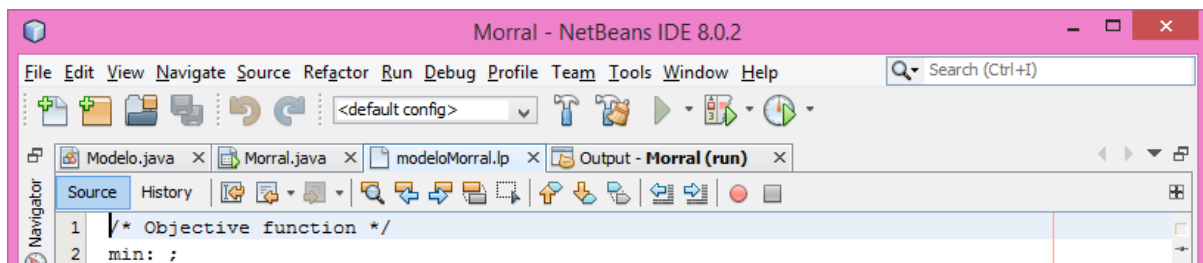
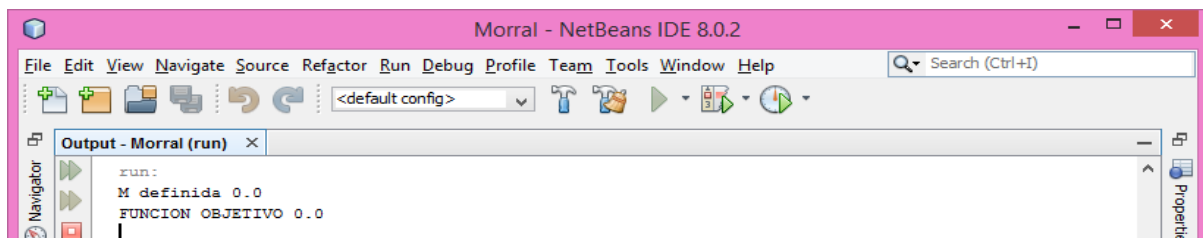


Prueba 7. Prueba de solución no factible con $n=0$

Otro ejemplo para una solución del modelo no factible, es cuando la cantidad de de cajas es 0.




No se construye el modelo y el valor de la Función Objetivo es 0.



Prueba 8. Prueba de n=20 valor de n grande y tiempo de ejecucion grande

Es una prueba con una cantidad de cajas considerable, si tomamos como referencia las pruebas anteriores. Además, con valores grandes para la capacidad del morral, volumen y peso. El programa y el modelo matemático se comportan correctamente al encontrar una solución óptima factible con 20 cajas, que son transportadas en 4 morrales. Los pesos y volúmenes de cada caja no exceden la capacidad máxima del morral, sin embargo alguno datos de peso se acercan del volumen se acercan a la capacidad máxima. Puede evidenciarse en la distribución de las cajas dentro de cada morral que la sumatoria de pesos y volúmenes de las cajas contenidas no excede la capacidad del morral. La solución es encontrada en un tiempo razonable de 10,17 ms teniendo en cuenta que el tamaño de las entrada.



Problema Morrales

Datos Cajas

Nº	Peso	Volumen
1	2	20
2	3	4
3	6	5
4	10	1
5	2	5
6	4	8
7	8	5
8	3	1
9	9	10

Datos Morral

Numero Cajas	Peso	Volumen
20.0	30	40

Distribución de las cajas en los morrales

Morral	Cajas
1	Caja 1;Caja 8;Caja 10;Caja 17;Caja 19;
2	Caja 4;Caja 6;Caja 9;Caja 11;Caja 12;
3	Caja 7;Caja 15;Caja 20;
4	Caja 2;Caja 3;Caja 5;Caja 13;Caja 14;Caja 16;Caja ...


Total minimo de personas (Total minimo de morrales)

Tiempo de ejecucion (ms)

Prueba 9. Prueba de solución en el peor caso donde n =cantidad mínima de morrales

El peor caso se presenta cuando la cantidad de morrales mínima es igual a la cantidad de n cajas en total, en este caso hay una caja en cada uno de los morrales. El programa y el modelo matemático se comportan correctamente al encontrar una solución óptima factible con 2 cajas, que son transportadas en 2 morrales. La solución es encontrada en un tiempo razonable de 0,043 ms teniendo en cuenta que la cantidad de cajas es pequeñas, el tiempo de ejecución no es elevado, aunque sea el peor caso.

Optimización Morrales

 **Problema Morrales**

C:\Users\Maleja\Documents\p9_Peorcaso.txt

Datos Cajas

N°	Peso	Volumen
1	2	5
2	1	4

Datos Morral

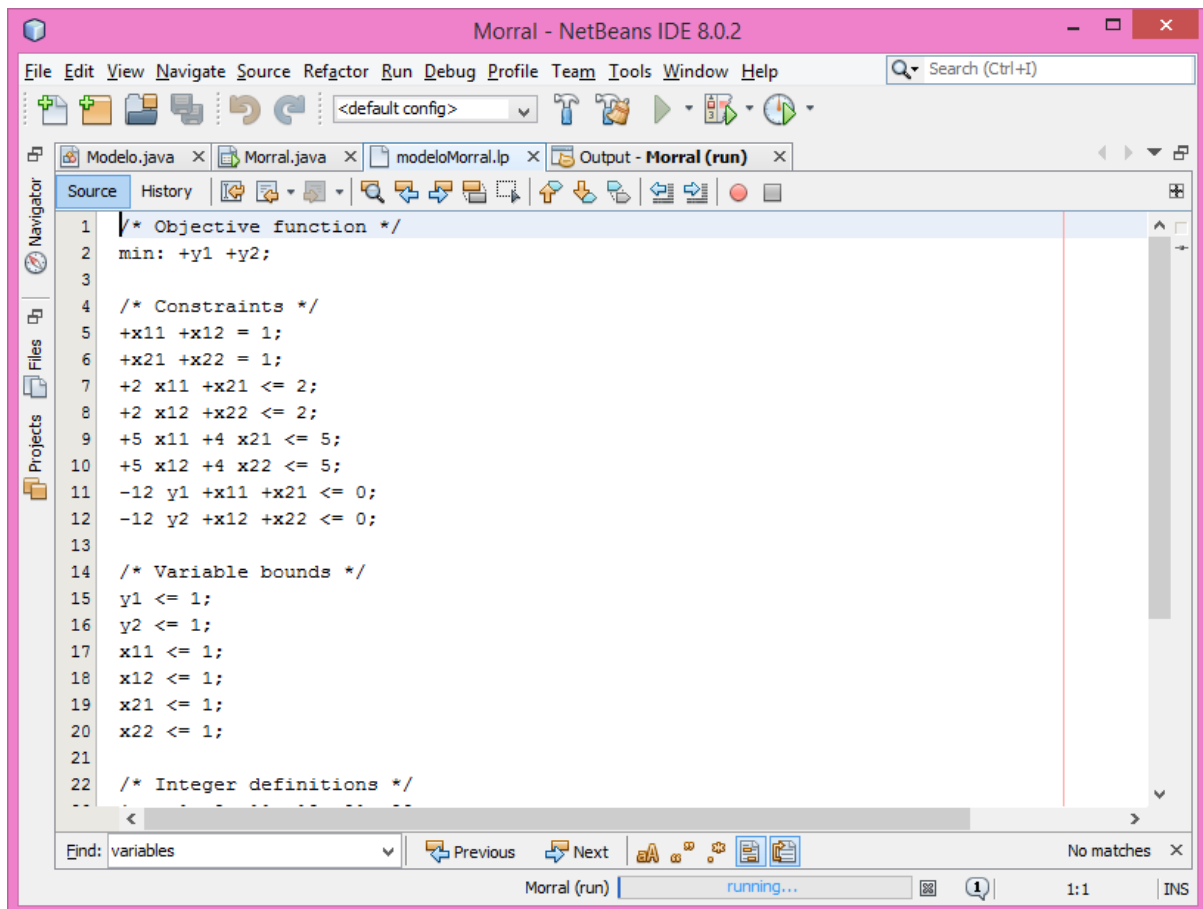
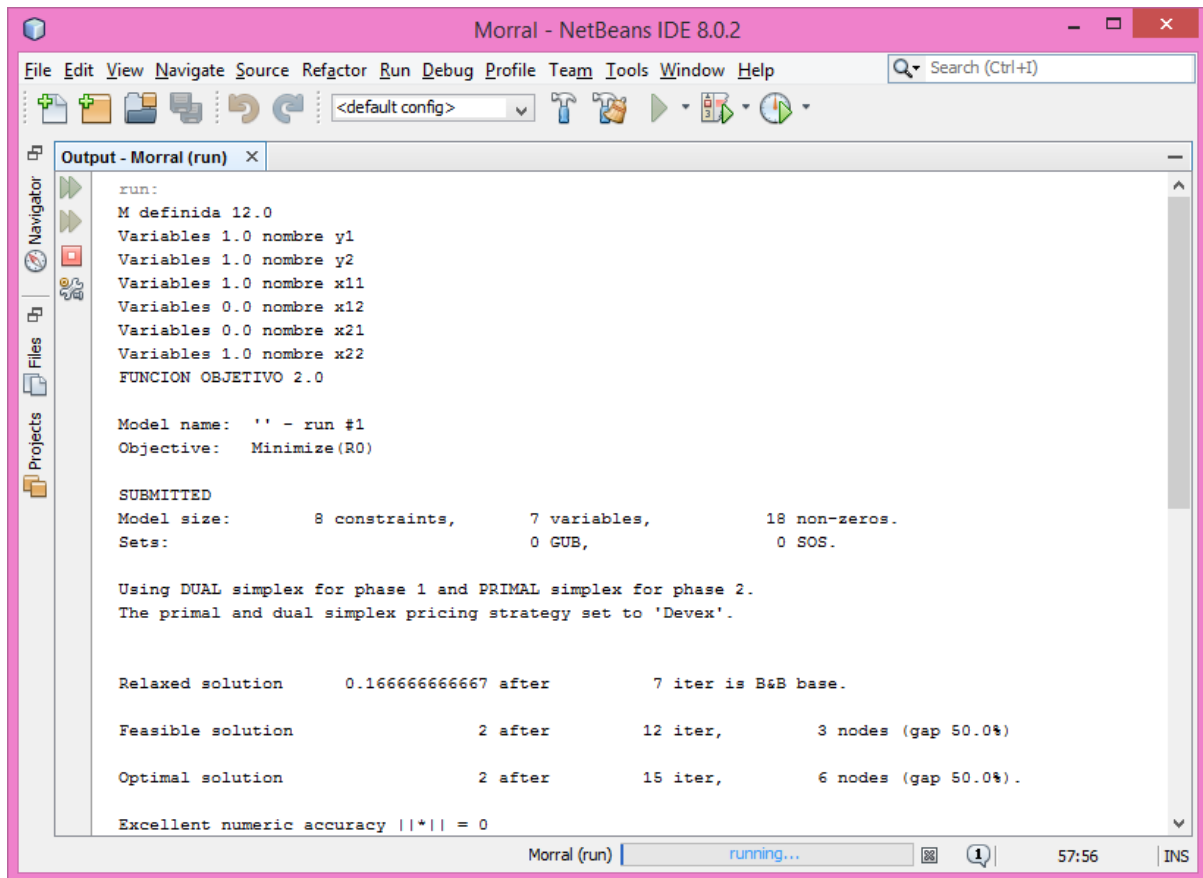
Numero Cajas	Peso	Volumen
2.0	2	5

Distribución de las cajas en los morrales

Morral	Cajas
1	Caja 1;
2	Caja 2;

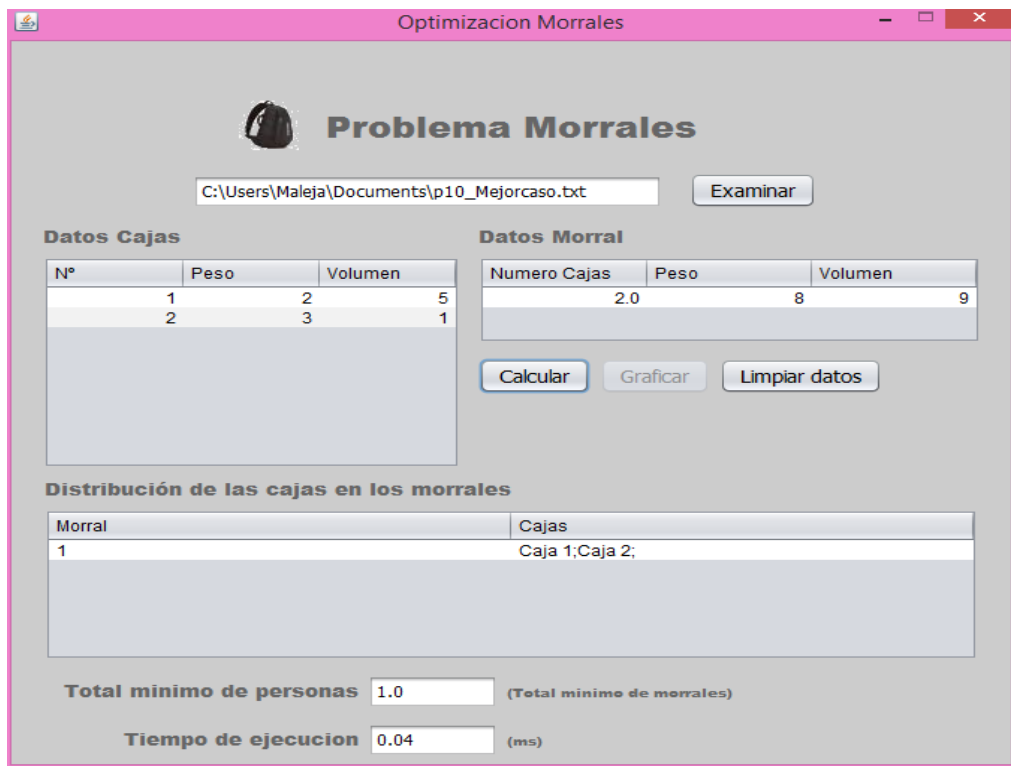
Total minimo de personas (Total minimo de morrales)

Tiempo de ejecucion (ms)



Prueba 10. Prueba de solución en el mejor caso donde cantidad de morrales es igual a 1

El mejor caso se presenta cuando toda la cantidad de cajas cabe en un solo morral. En esta prueba se encuentra una solución óptima factible con 2 cajas, que son transportadas en 1 morrales. El tiempo de ejecución es de 0,04 ms.



Optimizacion Morrales

Problema Morrales

C:\Users\Maleja\Documents\p10_Mejorcaso.txt **Examinar**

Datos Cajas

Nº	Peso	Volumen
1	2	5
2	3	1

Datos Morral

Numero Cajas	Peso	Volumen
2.0	8	9

Calcular **Graficar** **Limpiar datos**

Distribución de las cajas en los morrales

Morral	Cajas
1	Caja 1;Caja 2;

Total minimo de personas 1.0 (Total minimo de morrales)

Tiempo de ejecucion 0.04 (ms)

Morral - NetBeans IDE 8.0.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

Output - Morral (run)

```
run:
M definida 11.0
Variables 1.0 nombre y1
Variables 0.0 nombre y2
Variables 1.0 nombre x11
Variables 0.0 nombre x12
Variables 1.0 nombre x21
Variables 0.0 nombre x22
FUNCION OBJETIVO 1.0

Model name: '' - run #1
Objective: Minimize(R0)

SUBMITTED
Model size:      8 constraints,      7 variables,      18 non-zeros.
Sets:            0 GUB,              0 SOS.

Using DUAL simplex for phase 1 and PRIMAL simplex for phase 2.
The primal and dual simplex pricing strategy set to 'Devex'.

Relaxed solution      0.1818181818 after      6 iter is B&B base.

Feasible solution          1 after      7 iter,      1 nodes (gap 0.0%)

Optimal solution          1 after      7 iter,      1 nodes (gap 0.0%).

Excellent numeric accuracy ||*|| = 1.11022e-015
```

Morral (run) | running... | 57:56 | INS

Morral - NetBeans IDE 8.0.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

Modelo.java | Morral.java | modeloMorral.lp | Output - Morral (run)

Source | History

```
1  /* Objective function */
2  min: +y1 +y2;
3
4  /* Constraints */
5  +x11 +x12 = 1;
6  +x21 +x22 = 1;
7  +2 x11 +3 x21 <= 8;
8  +2 x12 +3 x22 <= 8;
9  +5 x11 +x21 <= 9;
10 +5 x12 +x22 <= 9;
11 -11 y1 +x11 +x21 <= 0;
12 -11 y2 +x12 +x22 <= 0;
13
14 /* Variable bounds */
15 y1 <= 1;
16 y2 <= 1;
17 x11 <= 1;
18 x12 <= 1;
19 x21 <= 1;
20 x22 <= 1;
21
22 /* Integer definitions */
23 int y1,y2,x11,x12,x21,x22;
24
```

Find: variables | Previous | Next | No matches

Morral (run) | running... | 1:1 | INS

4.1 PRUEBAS PARTE 2

Prueba 1. Mala distribución de cajas y al optimizar se igualan las cargas

The screenshot shows the 'Optimizacion Morrales' application window. The title bar reads 'Optimizacion Morrales'. The main title is 'Problema Morrales'. Below the title, there is a file path input field containing 'C:\Users\Maleja\Desktop\Pruebas\p1_pvpequenos.txt' and an 'Examinar' button.

There are two data input sections:

- Datos Cajas:** A table with 3 columns: 'Nº', 'Peso', and 'Volumen'. It contains 4 rows of data.

Nº	Peso	Volumen
1.0	2.0	1.0
2.0	1.0	1.0
3.0	2.0	2.0
4.0	1.0	3.0
- Datos Morral:** A table with 3 columns: 'Numero Cajas', 'Peso', and 'Volumen'. It contains 1 row of data.

Numero Cajas	Peso	Volumen
4.0	5.0	10.0

Below the input sections, there are buttons for 'Limpiar datos' and 'Calcular'. The 'Calcular' button has been clicked, resulting in the following output:

Distribución de las cajas en los morrales (Tiempo de ejecucion: 0.008s)

Morral	Cajas	Peso
1	Caja 2;Caja 3;Caja 4;	4
2	Caja 1;	2

Total minimo de personas: 2.0

Distribución equilibrada de las cajas en los morrales (Tiempo de ejecucion: 0.011s)

Morral	Cajas	Peso
1	Caja 3;Caja 4;	3
2	Caja 1;Caja 2;	3

En la prueba anterior se obtiene del Modelo 1, donde ingresan 4 cajas, que son empacadas en dos morrales con una mala distribución. Es un ejemplo sencillo donde al ejecutar el Modelo 2 al optimizar se terminan igualan las cargas, haciendo la máxima diferencia de los pesos igual a cero. De igual forma, sucede con la siguiente imagen donde se obtiene una distribución de cargas totalmente equitativa es fácil ver que el modelo funciona correctamente.

Optimizacion Morrales

Problema Morrales

Datos Cajas

Nº	Peso	Volumen
1.0	5.0	2.0
2.0	1.0	1.0
3.0	1.0	1.0
4.0	4.0	1.0
5.0	4.0	1.0

Datos Morral

Numero Cajas	Peso	Volumen
5.0	7.0	6.0

Distribución de las cajas en los morrales Tiempo de ejecucion 0.01s

Morral	Cajas	Peso
1	Caja 2;Caja 3;Caja 4;	6
2	Caja 5;	4
3	Caja 1;	5

Total minimo de personas

Distribución equilibrada de las cajas en los morrales Tiempo de ejecucion 0.0s

Morral	Cajas	Peso
1	Caja 1;	5
2	Caja 3;Caja 5;	5
3	Caja 2;Caja 4;	5

Prueba 2. Distribución de cargas donde no puede ser mejorada.

Optimizacion Morrales

Problema Morrales

Datos Cajas

Nº	Peso	Volumen
1.0	6.0	14.0
2.0	5.0	13.0
3.0	6.0	14.0
4.0	5.0	13.0
5.0	6.0	14.0
6.0	5.0	13.0

Datos Morral

Numero Cajas	Peso	Volumen
6.0	7.0	15.0

Distribución de las cajas en los morrales Tiempo de ejecucion 0.008s

Morral	Cajas	Peso
1	Caja 4;	5
2	Caja 6;	5
3	Caja 2;	5
4	Caja 5;	6
5	Caja 1;	6
6	Caja 3;	6

Total minimo de personas

Distribución equilibrada de las cajas en los morrales Tiempo de ejecucion 0.017s

Morral	Cajas	Peso
1	Caja 2;	5
2	Caja 6;	5
3	Caja 4;	5
4	Caja 5;	6
5	Caja 3;	6
6	Caja 1;	6

En este caso, que es un ejemplo del peor caso para el Modelo1, sirve para evidenciar que el modelo se comporta adecuadamente cuando las restricciones de volumen y peso no permiten hacer una mejor distribución. Como también sucede en la siguiente prueba:

Optimizacion Morrales

Problema Morrales

:\Users\Maleja\Desktop\Pruebas\p4_pvigualesaPV.txt

Datos Cajas

Nº	Peso	Volumen
1.0	6.0	15.0
2.0	4.0	1.0
3.0	4.0	2.0

Datos Morral

Numero Cajas	Peso	Volumen
3.0	6.0	15.0

Distribución de las cajas en los morrales Tiempo de ejecucion 0.008s

Morral	Cajas	Peso
1	Caja 1;	6
2	Caja 3;	4
3	Caja 2;	4

Total minimo de personas

Distribución equilibrada de las cajas en los morrales Tiempo de ejecucion 0.004s

Morral	Cajas	Peso
1	Caja 3;	4
2	Caja 1;	6
3	Caja 2;	4

Prueba 3. Caso donde se empaca la caja más grande un mortal y se distribuye la carga equilibradamente en el resto de cajas.

Optimizacion Morrales

Problema Morrales

:\No convienen\morrales iguales\p11_PesoMayor4.txt

Datos Cajas

Nº	Peso	Volumen
1.0	4.0	2.0
2.0	1.0	1.0
3.0	1.0	1.0
4.0	4.0	1.0
5.0	4.0	1.0

Datos Morral

Numero Cajas	Peso	Volumen
5.0	7.0	6.0

Distribución de las cajas en los morrales Tiempo de ejecucion 0.018s

Morral	Cajas	Peso
1	Caja 1;Caja 2;Caja 3;	6
2	Caja 5;	4
3	Caja 4;	4

Total minimo de personas

Distribución equilibrada de las cajas en los morrales Tiempo de ejecucion 0.005s

Morral	Cajas	Peso
1	Caja 1;Caja 3;	5
2	Caja 2;Caja 5;	5
3	Caja 4;	4

La prueba anterior y la siguiente son ejemplos básicos, variando el mayor peso con la misma cantidad de cajas y morrales, para evidenciar que el modelo y la aplicación se comportan bien para los casos donde la caja con mayor peso se empaca en un morral y se distribuyen equilibradamente las cargas en el resto de los morrales.

Optimizacion Morrales

Problema Morrales

\No convienen\morrales iguales\p11_PesoMayor7.txt

Datos Cajas

Nº	Peso	Volumen
3.0	1.0	1.0
1.0	7.0	2.0
2.0	1.0	1.0
3.0	1.0	1.0
4.0	4.0	1.0
5.0	4.0	1.0

Datos Morral

Numero Cajas	Peso	Volumen
5.0	7.0	6.0

Distribución de las cajas en los morrales Tiempo de ejecucion 0.009s

Morral	Cajas	Peso
1	Caja 4;	4
2	Caja 2;Caja 3;Caja 5;	6
3	Caja 1;	7

Total minimo de personas

Distribución equilibrada de las cajas en los morrales Tiempo de ejecucion 0.0s

Morral	Cajas	Peso
1	Caja 1;	7
2	Caja 3;Caja 5;	5
3	Caja 2;Caja 4;	5

Prueba 4. Caso de mayor número de cajas y pesos grandes donde se obtiene pases pases distribución

Optimizacion Morrales

Problema Morrales

J:\Users\Maleja\Desktop\Pruebas\p14_Distribucion4 .txt

Datos Cajas

N°	Peso	Volumen
1.0	3.0	2.0
2.0	2.0	4.0
3.0	1.0	1.0
4.0	2.0	2.0
5.0	1.0	2.0
6.0	16.0	1.0

Datos Morral

Numero Cajas	Peso	Volumen
10.0	18.0	16.0

Distribución de las cajas en los morrales Tiempo de ejecucion 0.085s

Morral	Cajas	Peso
1	Caja 7;Caja 8;	11
2	Caja 1;Caja 3;Caja 4;Caja 5;Caja 9;Caja 10;	15
3	Caja 2;Caja 6;	18

Total minimo de personas

Distribución equilibrada de las cajas en los morrales Tiempo de ejecucion 0.019s

Morral	Cajas	Peso
1	Caja 2;Caja 3;Caja 4;Caja 5;Caja 9;Caja 10;	14
2	Caja 1;Caja 7;Caja 8;	14
3	Caja 6;	16

Esta prueba es similar a la anterior pero con mayor cantidad de cajas y volúmenes más grandes donde se observa mejor el fea de cargas, si se fija la atención en la distribución arrojada por el modelo 1 y la obtenida finalmente por el modelo 2.

Prueba 5. Con 10 cajas y se obtiene una distribución equilibrada.

Optimizacion Morrales

Problema Morrales

C:\Users\Maleja\Desktop\Pruebas\p2_pvgrandes.txt Examinar

Datos Cajas

Nº	Peso	Volumen
1.0	2.0	20.0
2.0	3.0	4.0
3.0	25.0	5.0
4.0	10.0	1.0
5.0	13.0	5.0
6.0	4.0	10.0

Datos Morral

Numero Cajas	Peso	Volumen
10.0	25.0	30.0

Limpiar datos Calcular

Distribución de las cajas en los morrales Tiempo de ejecucion 0.144s

Morral	Cajas	Peso
1	Caja 5;Caja 6;Caja 7;	25
2	Caja 1;Caja 2;Caja 8;Caja 10;	21
3	Caja 4;Caja 9;	22
4	Caja 3;	25

Total minimo de personas

Distribución equilibrada de las cajas en los morrales Tiempo de ejecucion 0.02s

Morral	Cajas	Peso
1	Caja 2;Caja 6;Caja 9;Caja 10;	22
2	Caja 1;Caja 5;Caja 7;	23
3	Caja 3;	25
4	Caja 4;Caja 8;	23

La anterior prueba es una buena distribución, donde la máxima diferencia entre pesos del morral más vacío y el más lleno es 3, para un ejemplo de 10 cajas que inicialmente son empacados en 4 desequilibradamente.

A continuación se anexa el modelo para el ejemplo anterior que construye la aplicación para optimizar la distribución de las cajas:

```

3  /* Constraints */
4  -maxd +2 x11 -2 x12 +3 x21 -3 x22 +25 x31 -25 x32 +10 x41 -10 x42 +13 x51 -13 x52 +4 x61 -4 x62 +8 x71 -8 x72 +13 x81 -13 x82 +12 x91 -12 x
5  -maxd -2 x11 +2 x12 -3 x21 +3 x22 -25 x31 +25 x32 -10 x41 +10 x42 -13 x51 +13 x52 -4 x61 +4 x62 -8 x71 +8 x72 -13 x81 +13 x82 -12 x91 +12 x
6  -maxd +2 x11 -2 x13 +3 x21 -3 x23 +25 x31 -25 x33 +10 x41 -10 x43 +13 x51 -13 x53 +4 x61 -4 x63 +8 x71 -8 x73 +13 x81 -13 x83 +12 x91 -12 x
7  -maxd -2 x11 +2 x13 -3 x21 +3 x23 -25 x31 +25 x33 -10 x41 +10 x43 -13 x51 +13 x53 -4 x61 +4 x63 -8 x71 +8 x73 -13 x81 +13 x83 -12 x91 +12 x
8  -maxd +2 x11 -2 x14 +3 x21 -3 x24 +25 x31 -25 x34 +10 x41 -10 x44 +13 x51 -13 x54 +4 x61 -4 x64 +8 x71 -8 x74 +13 x81 -13 x84 +12 x91 -12 x
9  -maxd -2 x11 +2 x14 -3 x21 +3 x24 -25 x31 +25 x34 -10 x41 +10 x44 -13 x51 +13 x54 -4 x61 +4 x64 -8 x71 +8 x74 -13 x81 +13 x84 -12 x91 +12 x
10 -maxd +2 x12 -2 x13 +3 x22 -3 x23 +25 x32 -25 x33 +10 x42 -10 x43 +13 x52 -13 x53 +4 x62 -4 x63 +8 x72 -8 x73 +13 x82 -13 x83 +12 x92 -12 x
11 -maxd -2 x12 +2 x13 -3 x22 +3 x23 -25 x32 +25 x33 -10 x42 +10 x43 -13 x52 +13 x53 -4 x62 +4 x63 -8 x72 +8 x73 -13 x82 +13 x83 -12 x92 +12 x
12 -maxd +2 x12 -2 x14 +3 x22 -3 x24 +25 x32 -25 x34 +10 x42 -10 x44 +13 x52 -13 x54 +4 x62 -4 x64 +8 x72 -8 x74 +13 x82 -13 x84 +12 x92 -12 x
13 -maxd -2 x12 +2 x14 -3 x22 +3 x24 -25 x32 +25 x34 -10 x42 +10 x44 -13 x52 +13 x54 -4 x62 +4 x64 -8 x72 +8 x74 -13 x82 +13 x84 -12 x92 +12 x
14 -maxd +2 x13 -2 x14 +3 x23 -3 x24 +25 x33 -25 x34 +10 x43 -10 x44 +13 x53 -13 x54 +4 x63 -4 x64 +8 x73 -8 x74 +13 x83 -13 x84 +12 x93 -12 x
15 -maxd -2 x13 +2 x14 -3 x23 +3 x24 -25 x33 +25 x34 -10 x43 +10 x44 -13 x53 +13 x54 -4 x63 +4 x64 -8 x73 +8 x74 -13 x83 +13 x84 -12 x93 +12 x
16 -maxd -2 x13 +2 x14 -3 x23 +3 x24 -25 x33 +25 x34 -10 x43 +10 x44 -13 x53 +13 x54 -4 x63 +4 x64 -8 x73 +8 x74 -13 x83 +13 x84 -12 x93 +12 x
17 +x11 +x12 +x13 +x14 = 1;
18 +x21 +x22 +x23 +x24 = 1;
19 +x31 +x32 +x33 +x34 = 1;
20 +x41 +x42 +x43 +x44 = 1;
21 +x51 +x52 +x53 +x54 = 1;
22 +x61 +x62 +x63 +x64 = 1;
23 +x71 +x72 +x73 +x74 = 1;
24 +x81 +x82 +x83 +x84 = 1;
25 +x91 +x92 +x93 +x94 = 1;
26 +x101 +x102 +x103 +x104 = 1;
27 +2 x11 +3 x21 +25 x31 +10 x41 +13 x51 +4 x61 +8 x71 +13 x81 +12 x91 +3 x101 <= 25;
28 +2 x12 +3 x22 +25 x32 +10 x42 +13 x52 +4 x62 +8 x72 +13 x82 +12 x92 +3 x102 <= 25;
29 +2 x13 +3 x23 +25 x33 +10 x43 +13 x53 +4 x63 +8 x73 +13 x83 +12 x93 +3 x103 <= 25;
30 +2 x14 +3 x24 +25 x34 +10 x44 +13 x54 +4 x64 +8 x74 +13 x84 +12 x94 +3 x104 <= 25;
31 +20 x11 +4 x21 +5 x31 +x41 +5 x51 +10 x61 +5 x71 +x81 +10 x91 +x101 <= 30;
32 +20 x12 +4 x22 +5 x32 +x42 +5 x52 +10 x62 +5 x72 +x82 +10 x92 +x102 <= 30;
33 +20 x13 +4 x23 +5 x33 +x43 +5 x53 +10 x63 +5 x73 +x83 +10 x93 +x103 <= 30;

```

```

24 +x81 +x82 +x83 +x84 = 1;
25 +x91 +x92 +x93 +x94 = 1;
26 +x101 +x102 +x103 +x104 = 1;
27 +2 x11 +3 x21 +25 x31 +10 x41 +13 x51 +4 x61 +8 x71 +13 x81 +12 x91 +3 x101 <= 25;
28 +2 x12 +3 x22 +25 x32 +10 x42 +13 x52 +4 x62 +8 x72 +13 x82 +12 x92 +3 x102 <= 25;
29 +2 x13 +3 x23 +25 x33 +10 x43 +13 x53 +4 x63 +8 x73 +13 x83 +12 x93 +3 x103 <= 25;
30 +2 x14 +3 x24 +25 x34 +10 x44 +13 x54 +4 x64 +8 x74 +13 x84 +12 x94 +3 x104 <= 25;
31 +20 x11 +4 x21 +5 x31 +x41 +5 x51 +10 x61 +5 x71 +x81 +10 x91 +x101 <= 30;
32 +20 x12 +4 x22 +5 x32 +x42 +5 x52 +10 x62 +5 x72 +x82 +10 x92 +x102 <= 30;
33 +20 x13 +4 x23 +5 x33 +x43 +5 x53 +10 x63 +5 x73 +x83 +10 x93 +x103 <= 30;
34 +20 x14 +4 x24 +5 x34 +x44 +5 x54 +10 x64 +5 x74 +x84 +10 x94 +x104 <= 30;
35
36 /* Variable bounds */
37 x11 <= 1;
38 x12 <= 1;
39 x13 <= 1;
40 x14 <= 1;
41 x21 <= 1;
42 x22 <= 1;
43 x23 <= 1;
44 x24 <= 1;
45 x31 <= 1;
46 x32 <= 1;
47 x33 <= 1;
48 x34 <= 1;
49 x41 <= 1;
50 x42 <= 1;
51 x43 <= 1;
52 x44 <= 1;
53 x51 <= 1;
54 x52 <= 1;

```

```

55 x53 <= 1;
56 x54 <= 1;
57 x61 <= 1;
58 x62 <= 1;
59 x63 <= 1;
60 x64 <= 1;
61 x71 <= 1;
62 x72 <= 1;
63 x73 <= 1;
64 x74 <= 1;
65 x81 <= 1;
66 x82 <= 1;
67 x83 <= 1;
68 x84 <= 1;
69 x91 <= 1;
70 x92 <= 1;
71 x93 <= 1;
72 x94 <= 1;
73 x101 <= 1;
74 x102 <= 1;
75 x103 <= 1;
76 x104 <= 1;
77
78 /* Integer definitions */
79 int maxd, x11, x12, x13, x14, x21, x22, x23, x24, x31, x32, x33, x34, x41, x42, x43, x44, x51, x52, x53, x54, x61, x62, x63, x64, x71, x72, x73, x74, x81, x82, x83, x84, x91, x92, x93, x94, x101, x102, x103, x104;
80

```

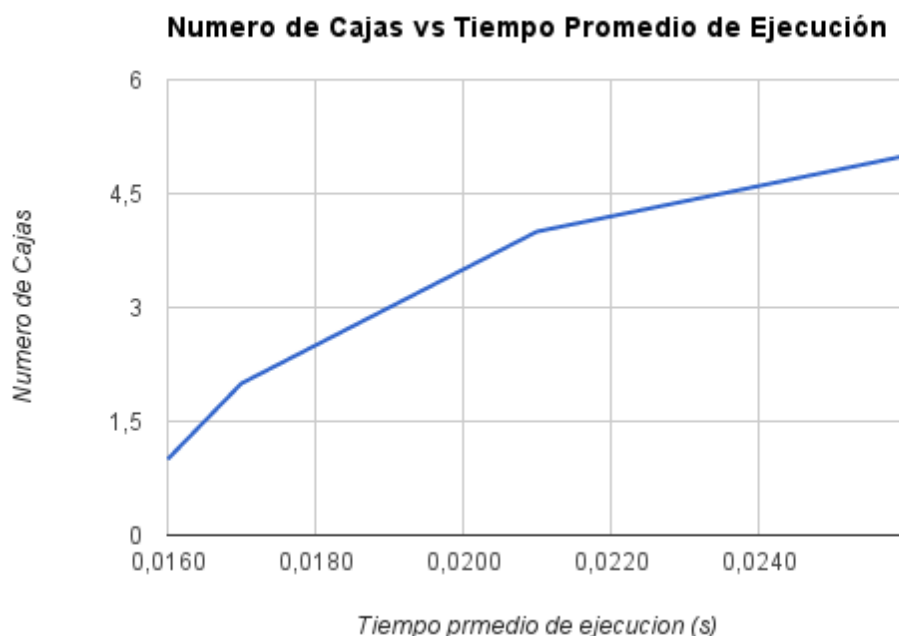
5. ANÁLISIS DE RESULTADOS

Tabla 7. Número de cajas y tiempos promedios de ejecución del programa completo para 5 ejecuciones

Número de cajas	Promedio tiempo ejecución (s)
1	0,0160
2	0,0170
3	0,0190

4	0,0210
5	0,0260

Gráfica 1. Número de cajas y tiempos promedios de ejecución del programa completo

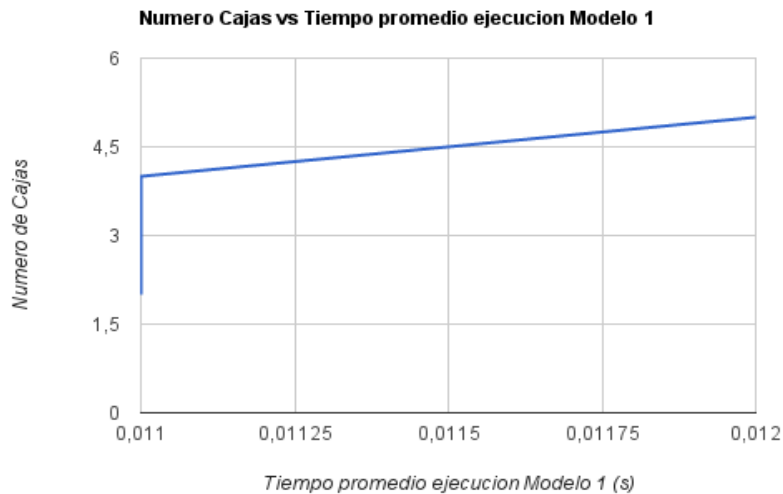


Para realizar el análisis del el tiempo de ejecución del programa que incluye creación del modelo, ejecución del modelo, gestión con interfaz grafica, y demas detalles que suceden dentro del modelo, se tuvo en cuenta el tamaño de las entradas cantidad de cajas y la aproximación de los pesos y volúmenes de cada caja a la capacidad máxima del morral. Según las pruebas realizadas el tiempo de ejecución del programa se ve afectado notablemente con el incremento del número total cantidad de cajas, en este caso se registró el tiempo para diferentes cantidad de cajas, sin variar la capacidad del morral (P y V) y las características de cada caja (peso- p_i y volumen v_i). Por lo que se puede observar que los diferentes métodos implementados tanto para la creacion y ejecucion del modelo como para la carga de archivo y carga de datos en la interfaz interviene en el incremento del tiempo de ejecución del programa a medida que la entrada n también crece.

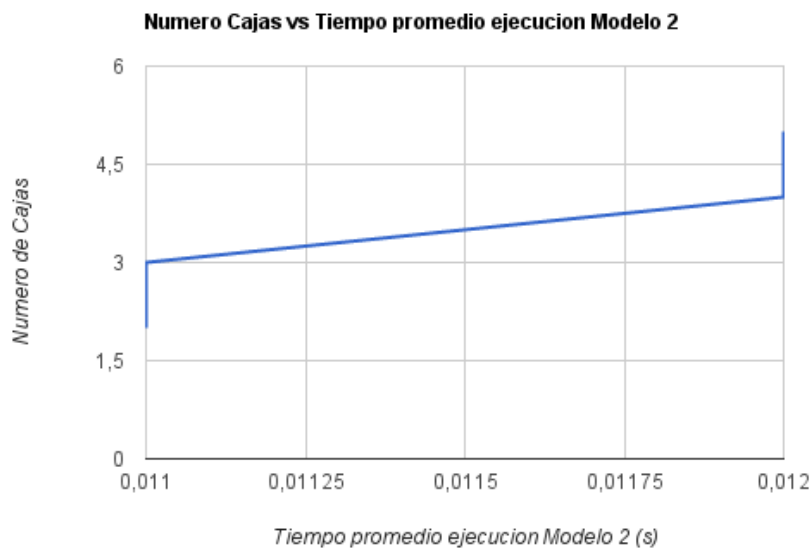
Tabla 8. Número de cajas y tiempos promedio de ejecución de los modelos para 5 ejecuciones

Número de cajas	Tiempo promedio ejecución Modelo 1 (s)	Tiempo promedio ejecución Modelo 2 (s)
1	0,012	0,012
2	0,011	0,011
3	0,011	0,012
4	0,011	0,011
5	0,012	0,012

Gráfica 2. Número de cajas y tiempos promedio de ejecución Modelo 1



Gráfica 3. Número de cajas y tiempos promedio de ejecución Modelo 2



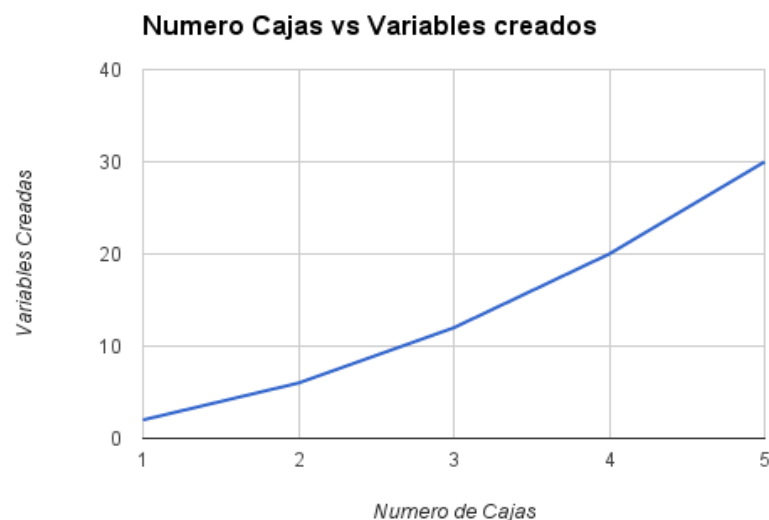
Para realizar el análisis del tiempo de ejecución de los modelos (al menos para los n pequeños) se debe tener en cuenta el tamaño de las entradas cantidad de cajas y la aproximación de los pesos y volúmenes de cada caja a la capacidad máxima del morral. Según las pruebas realizadas el tiempo de ejecución de los modelos se ve afectado muy notablemente con el incremento del número total cantidad de cajas cuando se compara un n muy pequeño con un n muy grande, en este caso se registró el tiempo para diferentes cantidad de cajas, sin variar la capacidad del morral (P y V) y las características de cada caja (peso- p_i y volumen v_i). Se puede afirmar, a modo de conclusión, que el tiempo de ejecución del modelo se ve afectado por la cantidad total de cajas cuando el n es considerablemente grande por ejemplo $n=20$, tanto en el modelo 1 como en el modelo 2, manteniendo las características de las cajas y los morrales iguales. Pero se puede

reflejar que para n parecidos como los que muestra la tabla el tiempo de ejecución tiende a ser constante y luego lineal, donde se refleja que el tiempo de solución del problema tiene a ser constante. Se destaca también que cuando el n es muy grande por ejemplo $n=20$, el modelo 1 se demora cierto tiempo pero el modelo 2 no logra terminar su ejecución, por esto se puede observar que el modelo 2 implica más tiempo de ejecución.

Tabla 9. Número de cajas vs cantidad de variables creadas Modelo 1

Número de cajas	Variables creadas Modelo 1
1	2
2	6
3	12
4	20
5	30

Gráfica 4. Número de cajas vs cantidad de variables creadas Modelo 1



En el análisis realizado para la mismas características de las cajas y la misma capacidad del morrales se observa que la creación de las variables tiene un comportamiento creciente que hace que la construcción del modelo sea más demorado y de igual forma, su procesamiento. El total de variables creadas en el modelo 1 está dada por la siguiente fórmula: número de cajas * (número de cajas + número de cajas).

Tabla 10. Número de morrales vs cantidad de variables creadas Modelo 2

Número de morrales	Variables creadas Modelo 2
1	2
2	5
3	10

4	17
5	26

Gráfica 5. Número de morrales vs cantidad de variables creadas Modelo 2

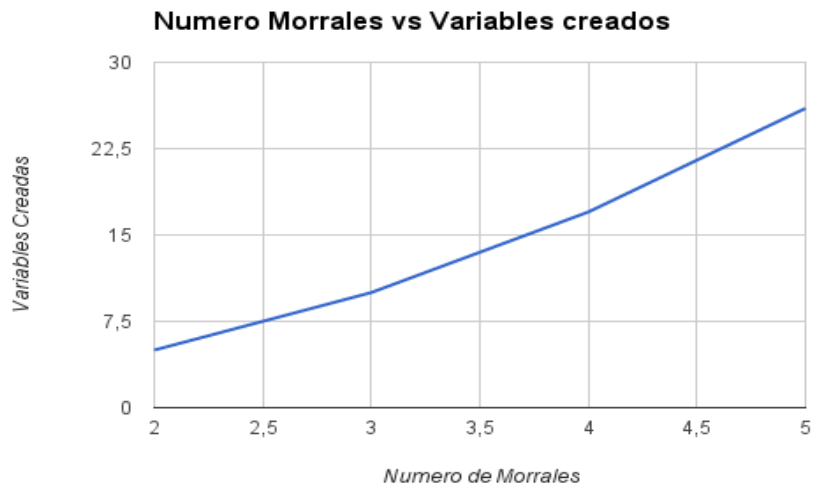
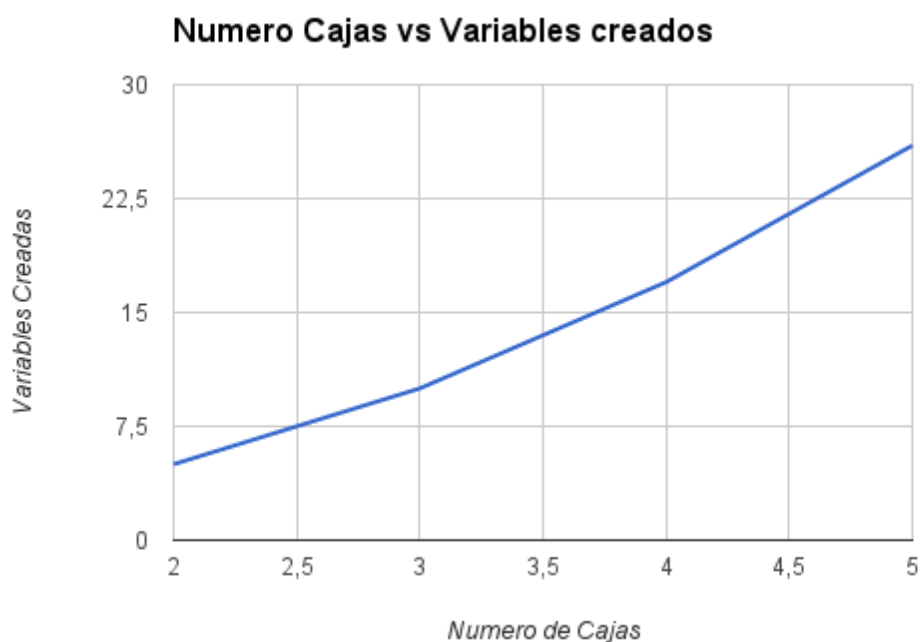


Tabla 11. Número de cajas vs cantidad de variables creadas Modelo 2

Número de cajas	Variables creadas Modelo 2
1	2
2	5
3	10
4	17
5	26

Gráfica 6. Número de cajas vs cantidad de variables creadas Modelo 2



En el análisis realizado para la mismas características de las cajas y la misma capacidad del morrales se observa que la creación de las variables tiene un comportamiento creciente que hace que la construcción del modelo sea más demorado y de igual forma, su procesamiento. El total de variables creadas en el modelo 2 está dada por la siguiente fórmula: $1 + (\text{número de cajas} * \text{número de morrales})$. Por lo que se puede decir que la cantidad de variables creadas en el modelo 2 dependen tanto del número de cajas como del número de morrales, a diferencia del modelo 1 donde solo depende del número de cajas.

Tabla 12. Número de cajas vs cantidad de restricciones creadas cuando m=1

Número de cajas	Restricciones creadas Modelo 1	Restricciones creadas Modelo 2
1	4	3
2	8	4
3	12	5
4	16	6
5	20	7

Gráfica 7. Número de cajas vs cantidad de restricciones creadas Modelo 1



Gráfica 8. Número de cajas vs cantidad de restricciones creadas Modelo 2



En el análisis realizado para la mismas características de las cajas y la misma capacidad del morrales se observa que para el modelo 1 y modelo 2 la creación de restricciones tiene un comportamiento creciente que hace que la construcción y procesamiento del modelo sea más demorado. El total de restricciones creadas en el modelo 1 está dada por la siguiente fórmula: número de cajas * 4. En cambio, el total de restricciones creadas en el modelo 2 depende de varios factores(cantidad de cajas y cantidad de morales) entre mas morrales y mas cajas sean entonces mayor es el número de restricciones y grande en comparación a la cantidad de restricciones del modelo 1; solo cuando la cantidad de morrales es $m=1$ (caso de la tabla 12) la cantidad de restricciones del modelo 2 es menor a la cantidad de restricciones del modelo 1, pues en este caso el modelo 2 no hace la restricción de la máxima diferencia porque solo hay 1 morral al cual calcularle su peso y no hay otro para comparar entre diferencia de pesos, entonces en este caso el modelo 2 podría ser más rápido en tiempos de ejecución que el modelo 1.

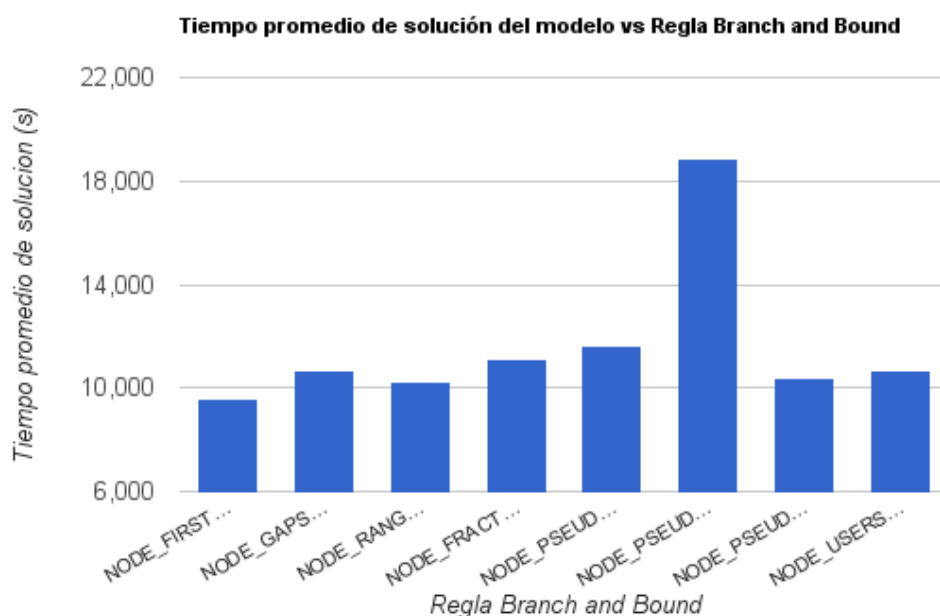
Mejora del Branch and Bound para reducir el tiempo de ejecución del modelo 1

Tabla 13. Regla Branch and Bound vs Tiempo promedio de solución del modelo 1
Teniendo en cuenta el caso de $n=20$ para el modelo 1 en 5 ejecuciones

Regla Branch and Bound	Tiempo promedio de solución del modelo (s)
NODE_FIRSTSELECT	9,565
NODE_GAPSELECT	10,675
NODE_RANGESELECT	10,270
NODE_FRACTIONSELECT	11,118
NODE_PSEUDOCOSTSELECT	11,608
NODE_PSEUDONONINTSELECT	18,885
NODE_PSEUDORATIOSELECT	10,412
NODE_USERSELECT	10,694

Gráfica 9. Regla Branch and Bound vs Tiempo promedio de ejecución del modelo 1

Teniendo en cuenta el caso de $n=20$ para el modelo 1 en 5 ejecuciones



Se probaron varios parámetros para mejorar la forma en cómo se arma el árbol de “Branch and Bound” en 5 ejecuciones para $n=20$ con el modelo 1, comparándolo con tiempos de ejecución del lp_solve y de esta forma poder volver al modelo y al programa lo más eficiente posible. Según los resultados de la tabla y el gráfico anterior, el parámetro que genera menos tiempo de solución es NODE_FIRSTSELECT, entonces con la instrucción `solver.setBbRule(LpSolve.NODE_FIRSTSELECT)` se mejora el Branch and Bound, donde la función `setBbRule(regla)` especifica la regla para elegir qué variables decimales se va a seleccionar en el árbol la cual puede influir en el tiempo de solución del modelo. La regla NODE_FIRSTSELECT realiza la selección de acuerdo al no entero más pequeño. Hay que tener en cuenta que dependiendo del modelo la regla que se especifique puede ser mejor o no, para cada modelo hay una regla que influyen mejor en la mejora de los tiempos de solución.

Mejora del Branch and Bound para reducir el tiempo de ejecución del modelo 2 y el modelo 1

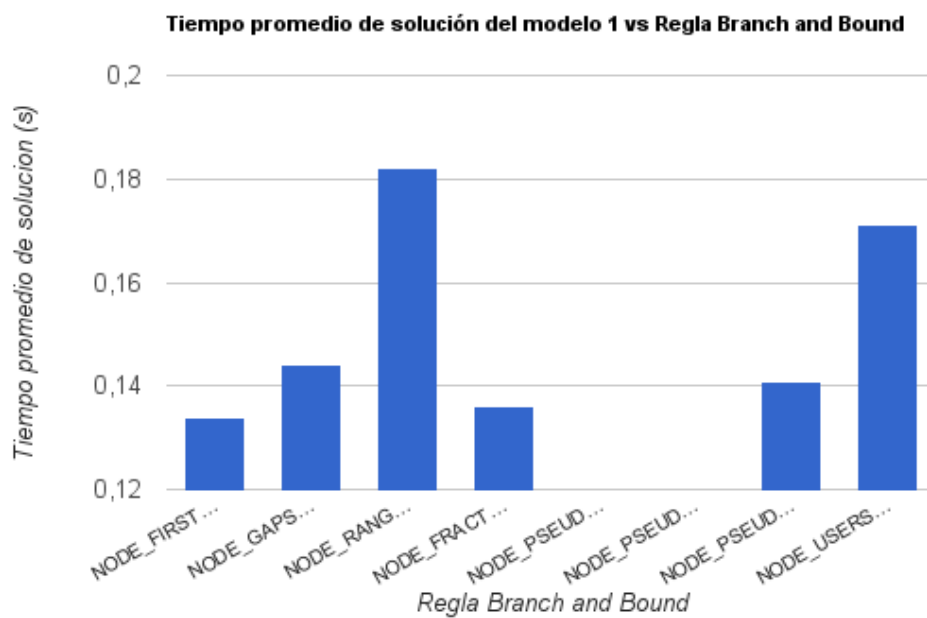
A continuación, se muestra una comparación al usar las mismas reglas de Branch and Bound para el modelo 1 y 2 vs tiempos promedio de ejecución para un $n=10$ y de acuerdo a esto se deja la mejor regla en la implementación:

Tabla 14. Regla Branch and Bound vs Tiempo promedio de ejecución del modelo 1 y Tiempo promedio de ejecución del modelo 2. Teniendo en cuenta el caso de $n=10$ para 5 ejecuciones

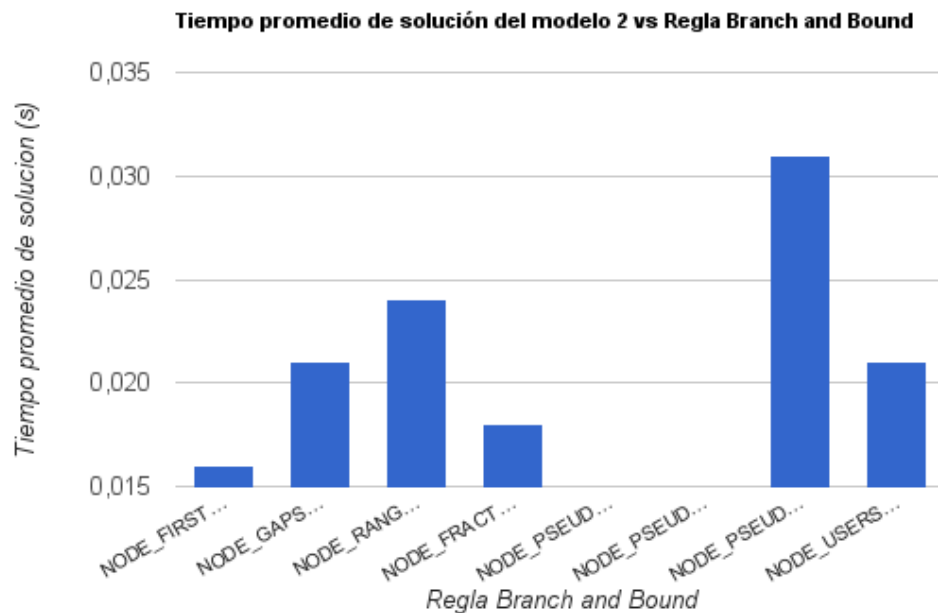
Regla Branch and Bound	Tiempo promedio de solución del modelo 2 (s)	Tiempo promedio de solución del modelo 1 (s)
------------------------	--	--

NODE_FIRSTSELECT	0,016	0,134
NODE_GAPSELECT	0,021	0,144
NODE_RANGESELECT	0,024	0,182
NODE_FRACTIONSELECT	0,018	0,136
NODE_PSEUDOCOSTSELECT	Se queda ejecutando	Se queda ejecutando
NODE_PSEUDONONINTSELECT	Se queda ejecutando	Se queda ejecutando
NODE_PSEUDORATIOSELECT	0,031	0,141
NODE_USERSELECT	0,021	0,171

Gráfica 10. Regla Branch and Bound vs Tiempo promedio de ejecución del modelo 1



Gráfica 11. Regla Branch and Bound vs Tiempo promedio de ejecución del modelo 2



Se probaron varios parámetros para mejorar la forma en cómo se arma el árbol de “Branch and Bound” en 5 ejecuciones para $n=10$ con el modelo 2 y con el modelo 2, comparándolo con tiempos de ejecución del `lp_solve` y de esta forma poder volver al modelo y al programa lo más eficiente posible. Según los resultados de la tabla y los gráficos anteriores, el parámetro que genera menos tiempo de solución sigue siendo `NODE_FIRSTSELECT`, entonces con la instrucción `solver.setBbRule(LpSolve.NODE_FIRSTSELECT)` se mejora el Branch and Bound tanto en la clase del modelo 1 como en la clase del modelo 2, donde la función `setBbRule(regla)` especifica la regla para elegir qué variables decimales se va a seleccionar en el árbol la cual puede influir en el tiempo de solución del modelo. Anteriormente se explicó cómo trabaja este parámetro.

De forma adicional, se esperaba que al insertar el modelo 2 la ejecución del programa fuera más eficiente entre más grande fuera el n , pero esto no fue así. Por ejemplo para el caso de $n=20$ cajas solo con el modelo 1 si se ejecutaba y mostraba un resultado, pero con el modelo y 2 el programa se queda procesando. Consideramos que esto puede ser causado por lo la adición de las restricciones que controlan lo de la máxima diferencia entre los pesos de los morrales, algo que no estaba en el modelo 1. Pero en términos de tiempos de ejecución del programa cuando n no es tan grande si mejora cuando se ejecuta con el modelo 2.

6. CONCLUSIONES

Se pudo comprobar, según los resultados obtenidos, que realizando una correcta modelación entre variables, función objetivo y restricciones, la programación lineal es muy útil para resolver problemas donde se requiere optimizar un valor de en determinada situación.

El primer modelo determina la solución óptima pero en el caso de la cantidad mínima de personas que deben llevar los morrales, esto es la cantidad mínima de morrales según una cantidad de cajas n . Sin embargo, en esta parte no se considera realizar la mejor distribución de las cajas en los morrales. Por esta razón, se propone el segundo modelo donde se mejora esta distribución de pesos.

Este problema de morrales no es un problema polinomial P , pero si es un problema NP , pues se puede verificar polinomialmente. Respecto a que sea un problema NPC no se puede saber, por lo que sería interesante demostrar que es NPC . Por esto es importante que se busque hacerlo lo más eficiente posible, y con la mejora del Branch and Bound que se realizó se logró realizar esto.