



# **Informe**

## **Proyecto Final**

### **Parte 1-Problema Morrales**

Presentado por:

María Alejandra Pabón 1310263

Mayerly Suarez 1310284

Complejidad y Optimización

Irene Tischer

# Informe Proyecto Final

## Parte 1-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:

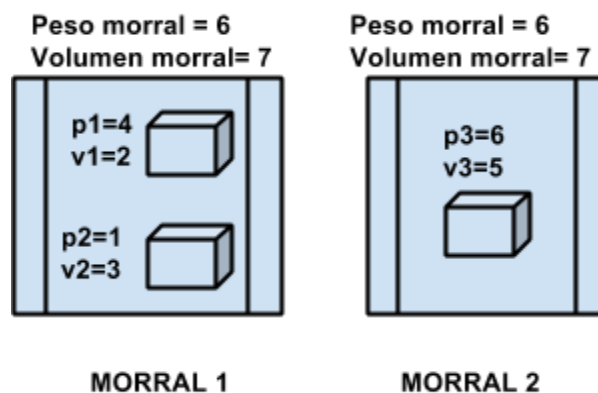


Gráfico 1. Ejemplo de solución óptima

### 2. DESCRIPCIÓN DEL MODELO

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:

## 2.1 Constantes

Tabla 1. Constantes

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.

## 2.2 Variables de decisión

Tabla 2. Variables de decisión

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

## 2.3 Función objetivo

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

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.

## 2.4 Restricciones

Tabla 3. Restricciones

Nº	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.

### 3. DETALLES DE IMPLEMENTACIÓN

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 y Morral.java. Modelo.java es la encargada de la creación del modelo (restricciones, función objetivo, modelar, nombres de variables y clasificación de las cajas en los morrales).

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 que procesa LPSolve, que se trabaja con variables binarias  $X_{ij}$  y  $Y_j$  se creó el método declararBinarias().

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
pesoMaximoMorral
volumenMaximoMorral
NumeroCaja PesoCaja Volumen Caja
NumeroCaja PesoCaja Volumen Caja
...
```

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 trabajó en la clase Modelo.java y se usaron los métodos nombrarVariables(), declararBinarias(), armarFuncionObjetivo(), armarPrimeraRestriccion(), armarSegundaRestriccion(), 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 las 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 modeloMorral.lp correspondiente al modelo que se construye, éste se crea dentro de la carpeta del proyecto /src/morral, en este archivo se puede verificar que restricciones y función objetivo se crearon.

También se creó el método `clasificarCajas(variablesdelmodelocreadas, resultadofuncionobjetivo)` para determinar la distribución de cajas en los morrales y así poder mostrar que las cajas deben ir finalmente en cada morral.

La  $M$  (grande) es una variable que ayuda a determinar si el morral no está vacío, como se explica en la Restricción 4 (Ver tabla de restricciones). 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. 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 las restricciones, 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)`.

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. Los tiempos de ejecución que propusimos calcular incluyen lo que se demora en construir el modelo y en resolverlo.

#### 4. PRUEBAS

**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". It contains a file path "C:\Users\Maleja\Documents\p1\_pvpequenos.txt" and an "Examinar" button. Below this are two data entry 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 buttons for "Calcular", "Graficar", and "Limpiar datos".

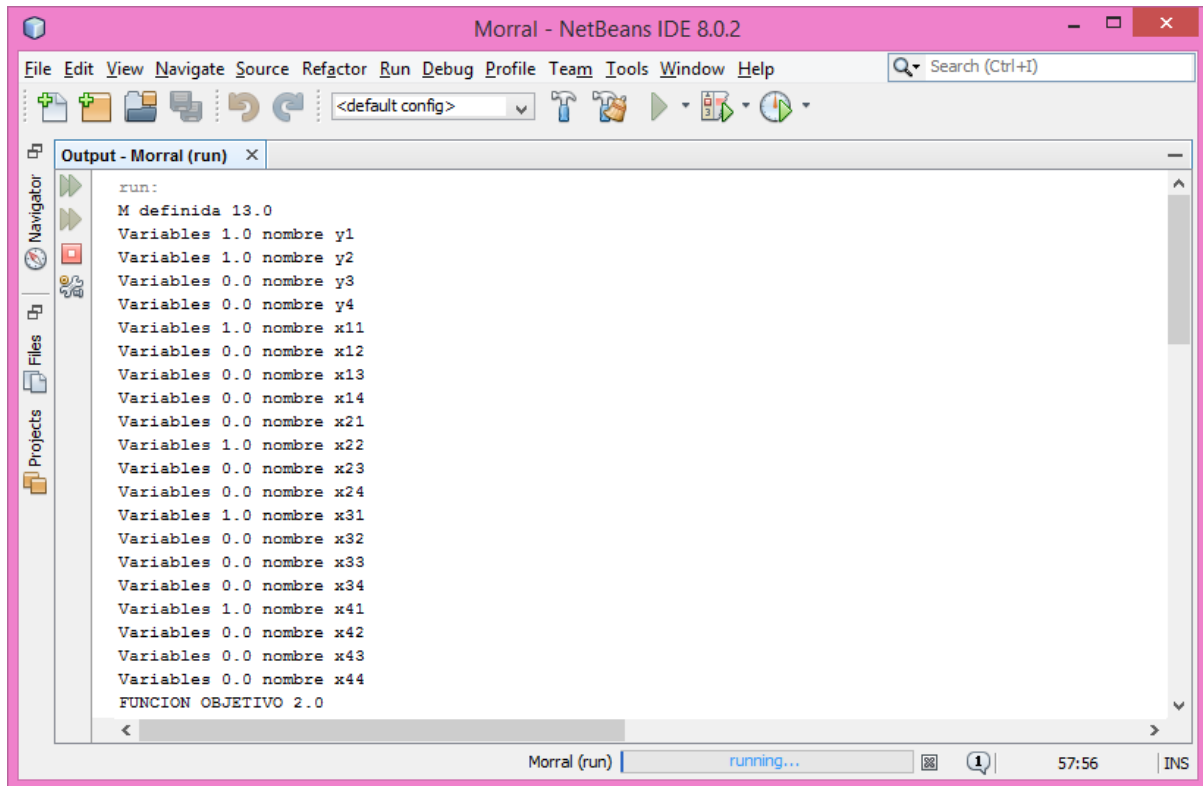
**Distribución de las cajas en los morrales**

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

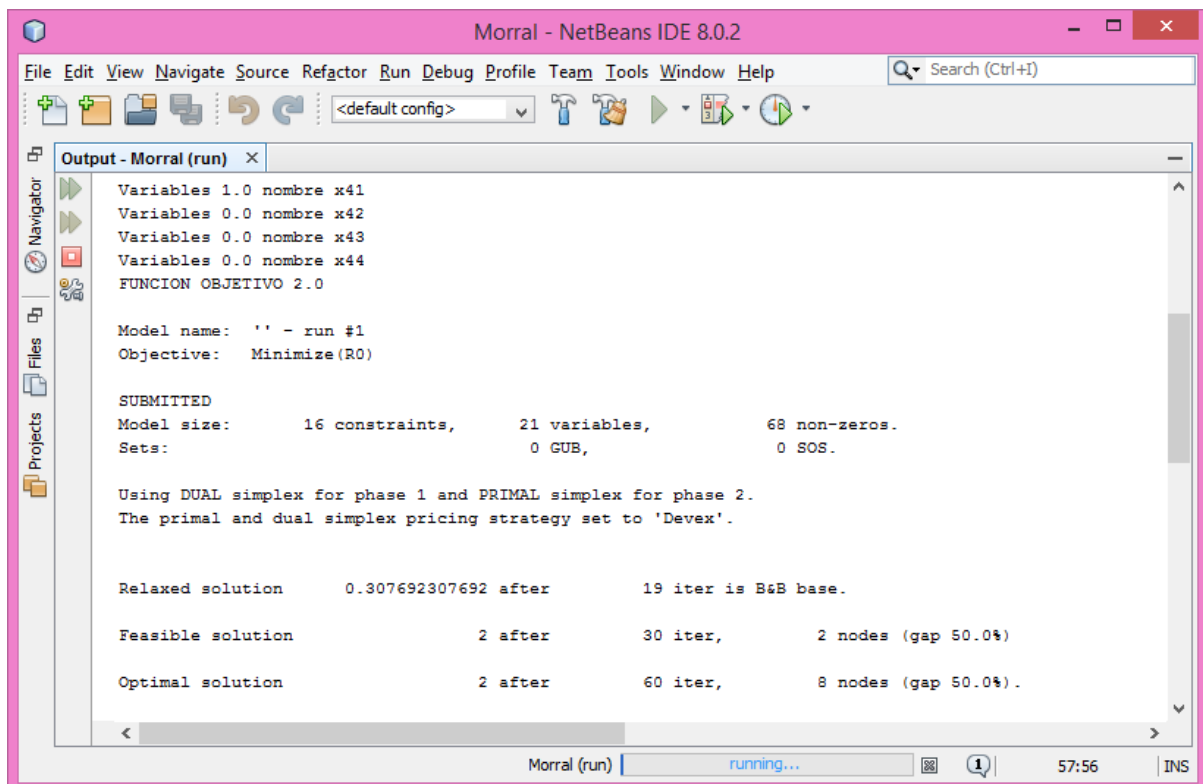
At the bottom, there are two status fields: "Total minimo de personas" with value "2.0" (Total minimo de morrales) and "Tiempo de ejecucion" with value "0.059" (ms).

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**

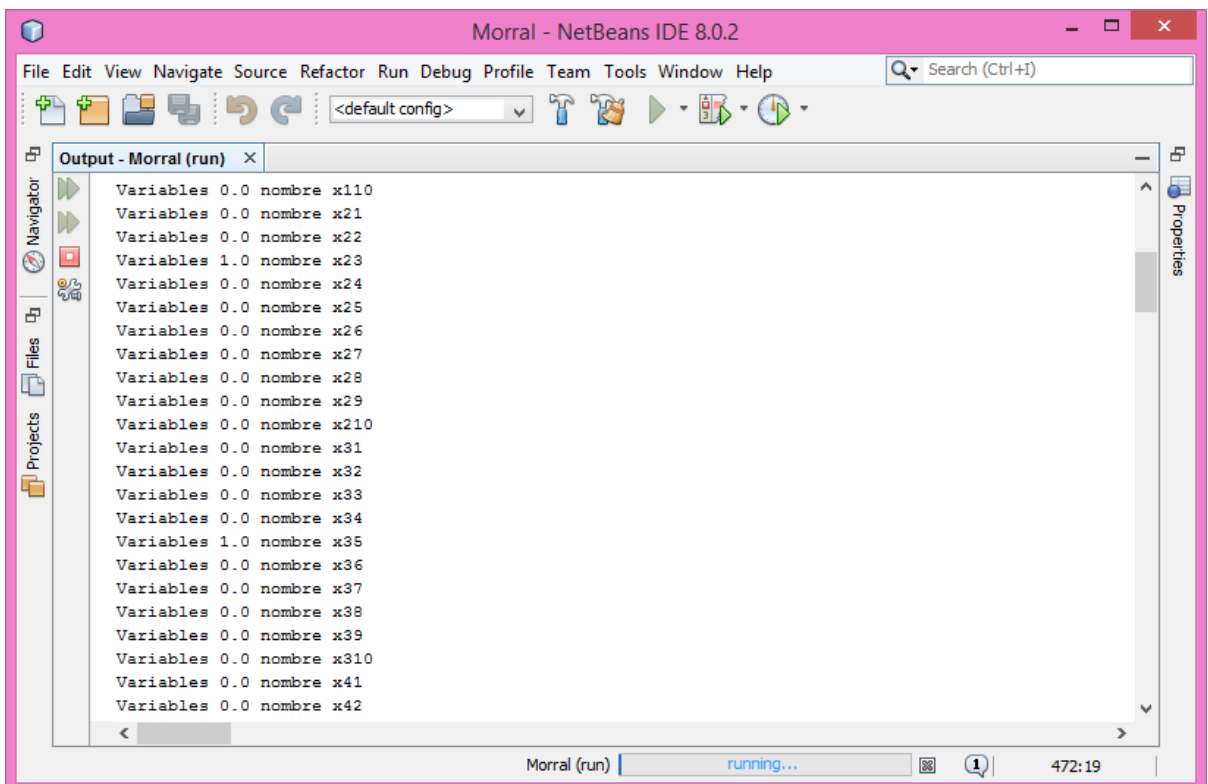
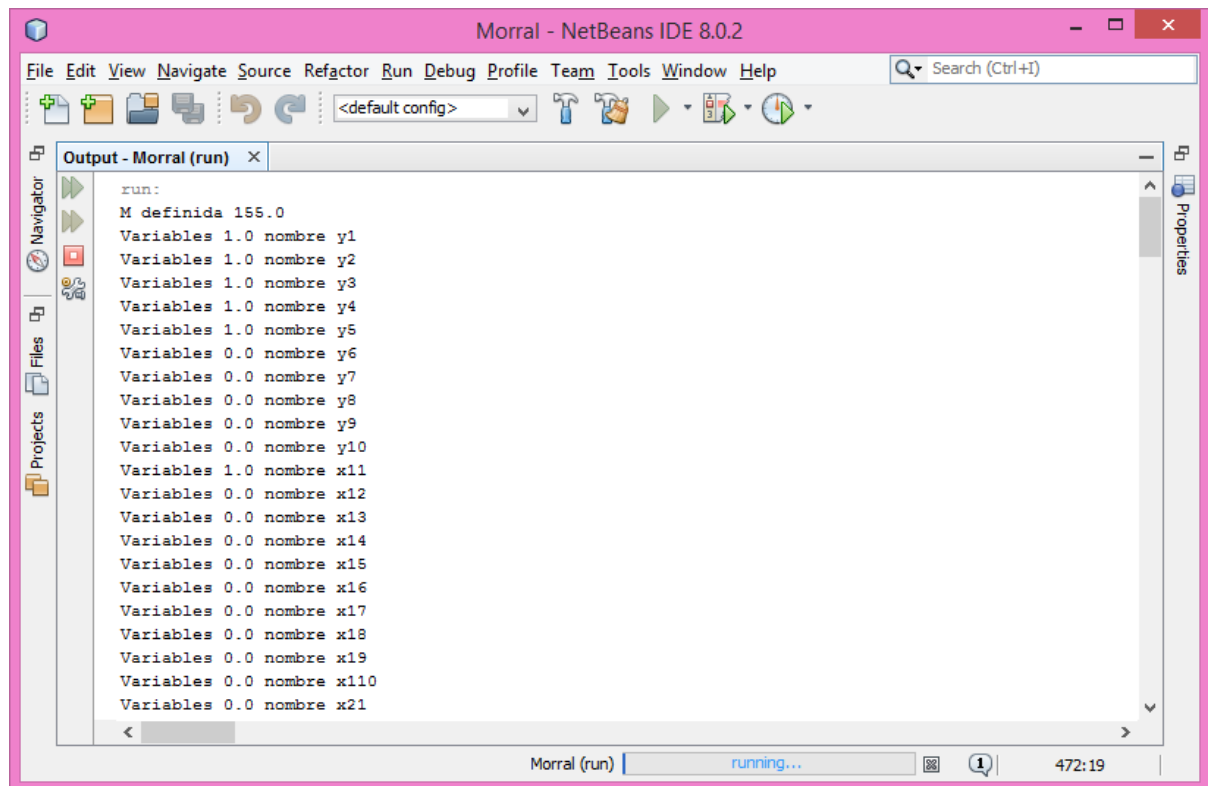
Morral	Cajas
1	Caja 1;
2	Caja 5;Caja 9;
3	Caja 2;Caja 6;Caja 7;Caja 10;
4	Caja 4;Caja 8;
5	Caja 3;

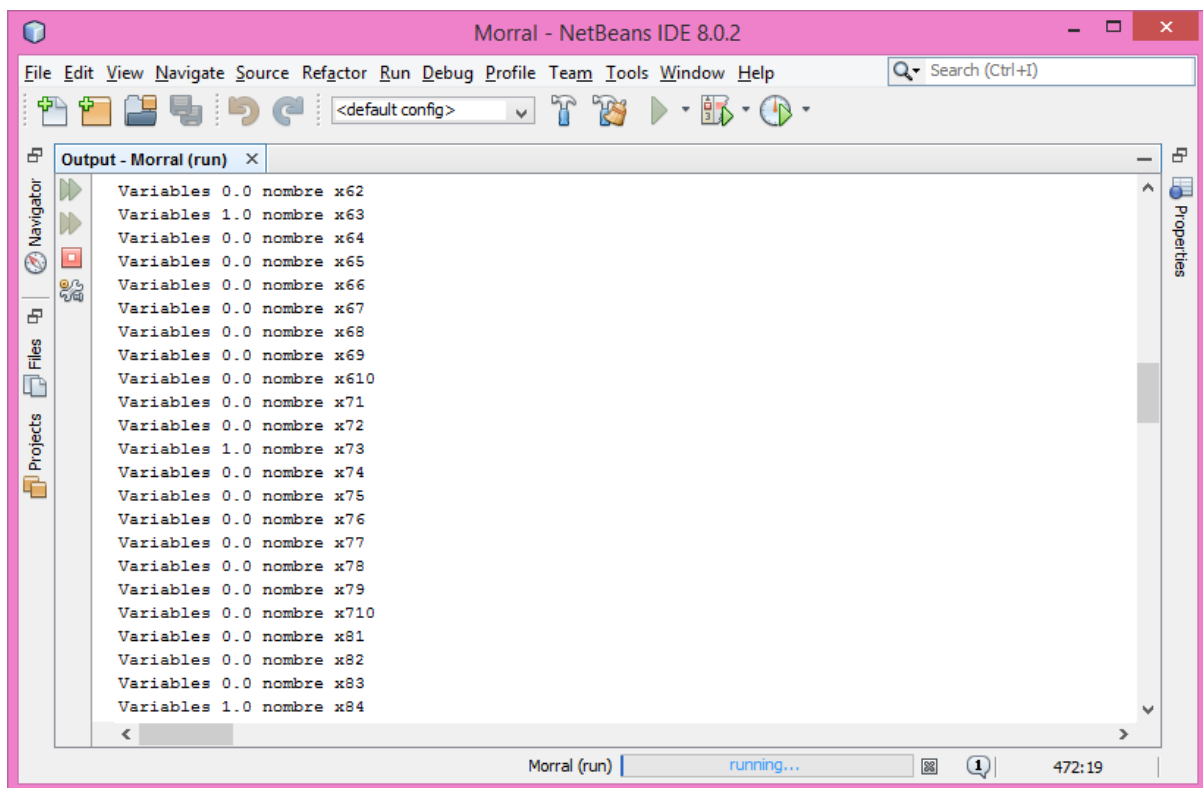
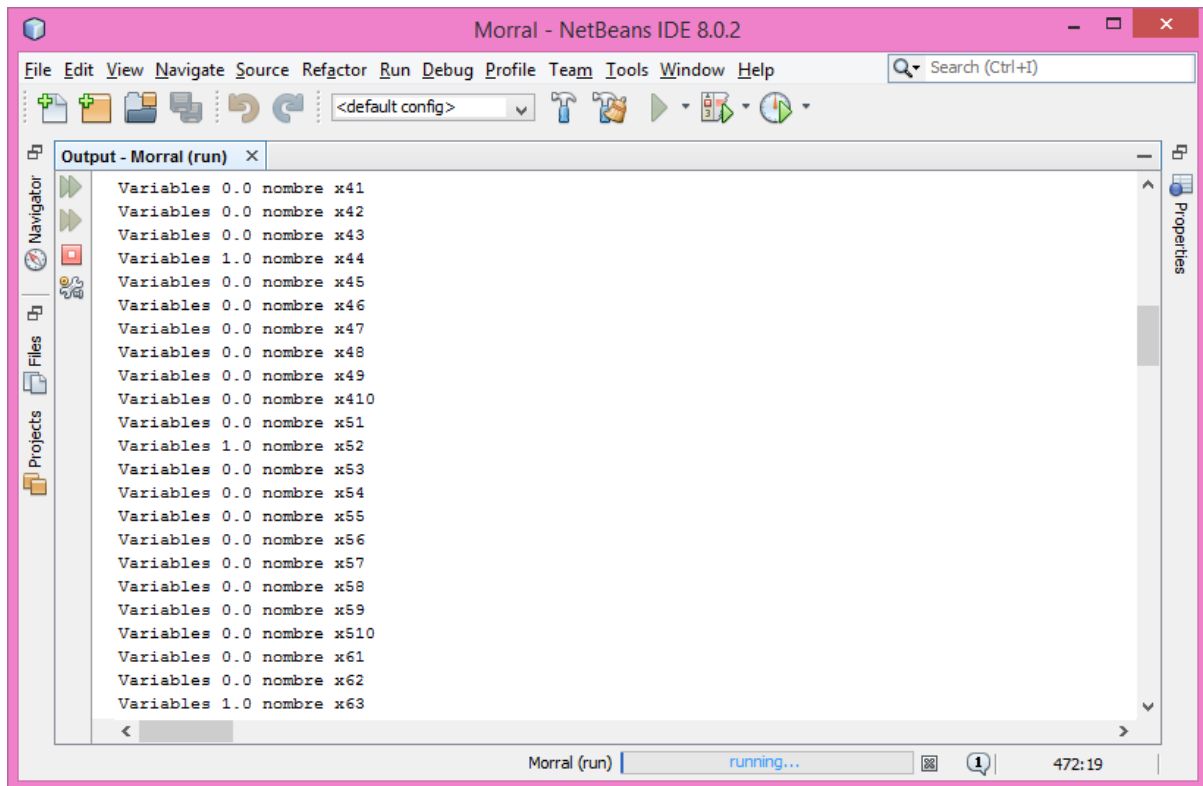
**Total minimo de personas** 5.0 (Total minimo de morrales)

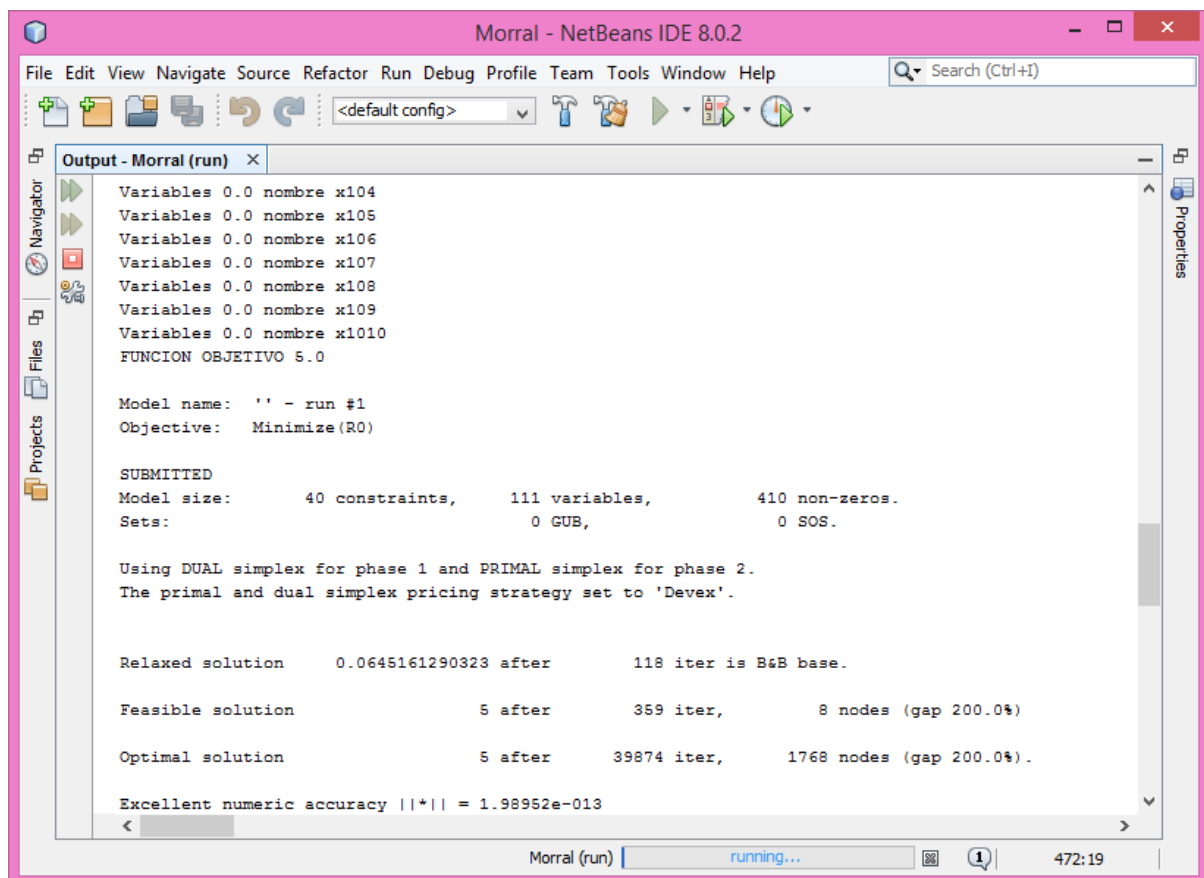
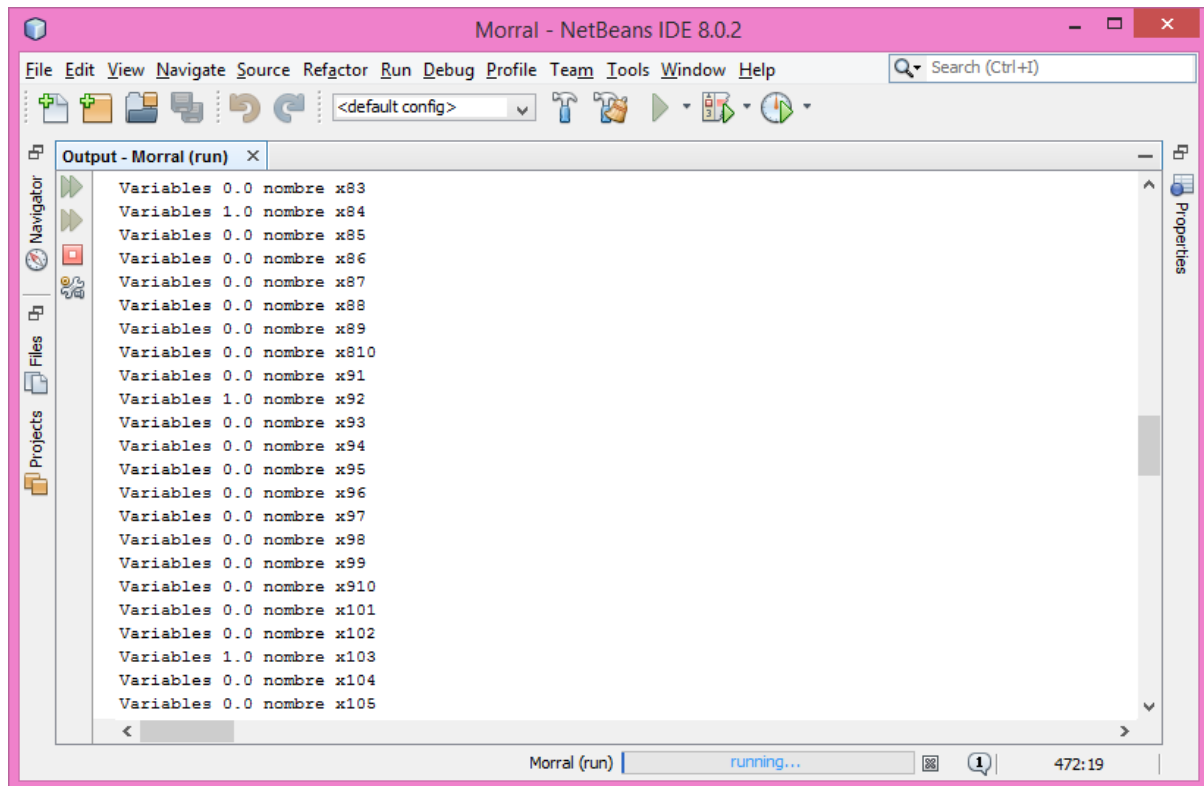
**Tiempo de ejecucion** 1.291 (ms)

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

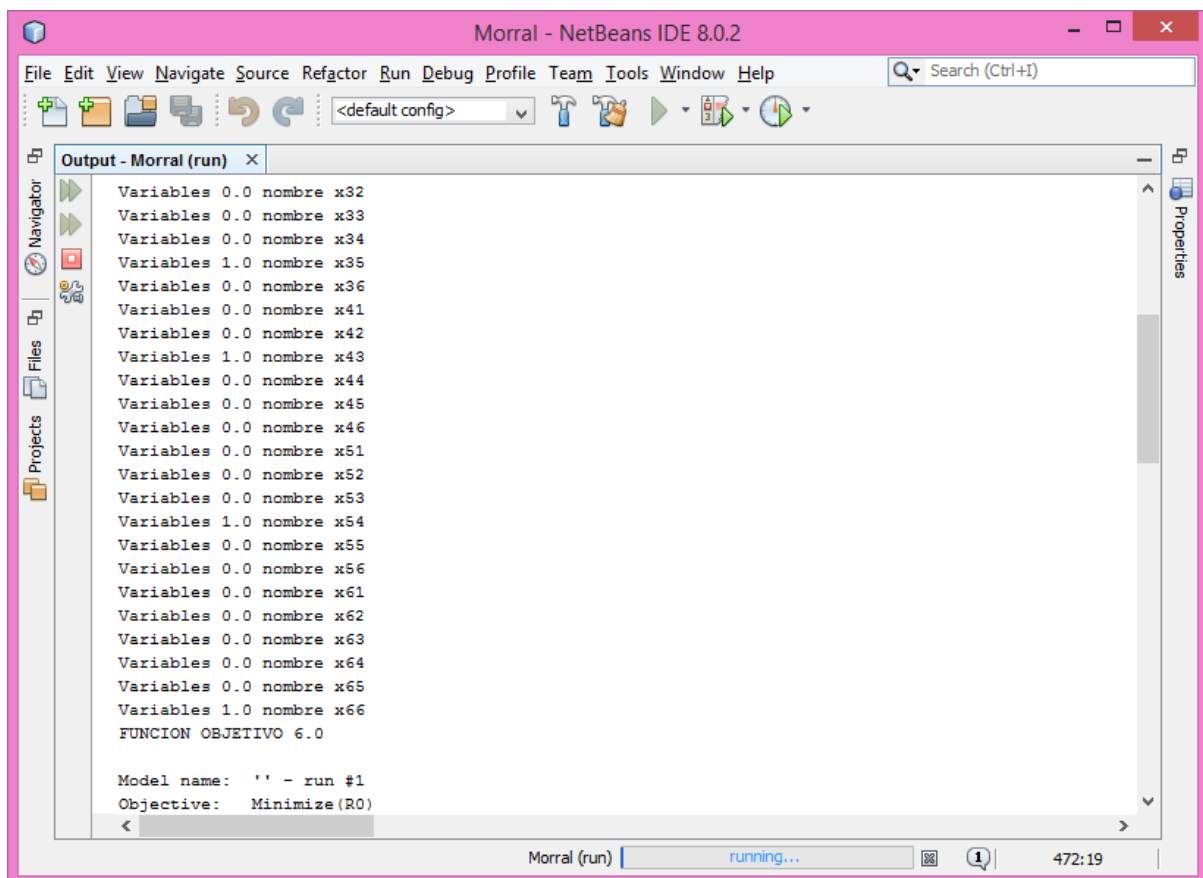
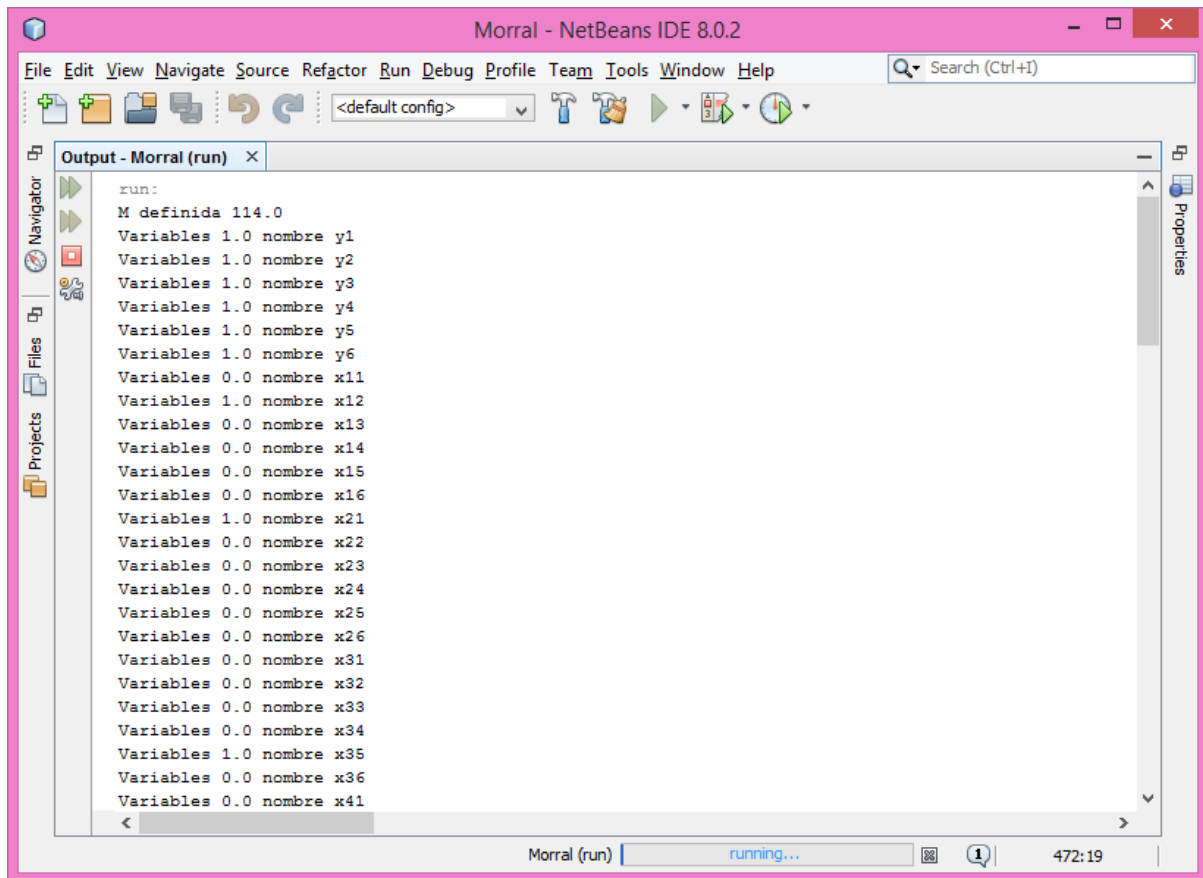
**Distribución de las cajas en los morrales**

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

**Datos Cajas**

N°	Peso	Volumen
1	6	15
2	4	1
3	4	1

**Datos Morral**

Numero Cajas	Peso	Volumen
3.0	6	15

**Distribución de las cajas en los morrales**

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

**Total minimo de personas**  (Total minimo de morrales)

**Tiempo de ejecucion**  (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 next to it. The main area is divided into two sections: 'Datos Cajas' and 'Datos Morral'. 'Datos Cajas' contains a table with 4 rows of box data. 'Datos Morral' contains a table with 1 row of knapsack data. Below these tables are buttons for 'Calcular', 'Graficar', and 'Limpiar datos'. A modal dialog box titled 'Mensaje' is open in the center, displaying an information icon and the text 'La solucion no es factible', with an 'Aceptar' button. At the bottom of the window, there are input fields for 'Total minimo de personas' and 'Tiempo de ejecucion', each followed by a label in parentheses: '(Total minimo de morrales)' and '(ms)' respectively.

N°	Peso	Volumen
1	8	1
2	5	4
3	17	1
4	5	4

Numero Cajas	Peso	Volumen
4.0	6	15

**Mensaje**

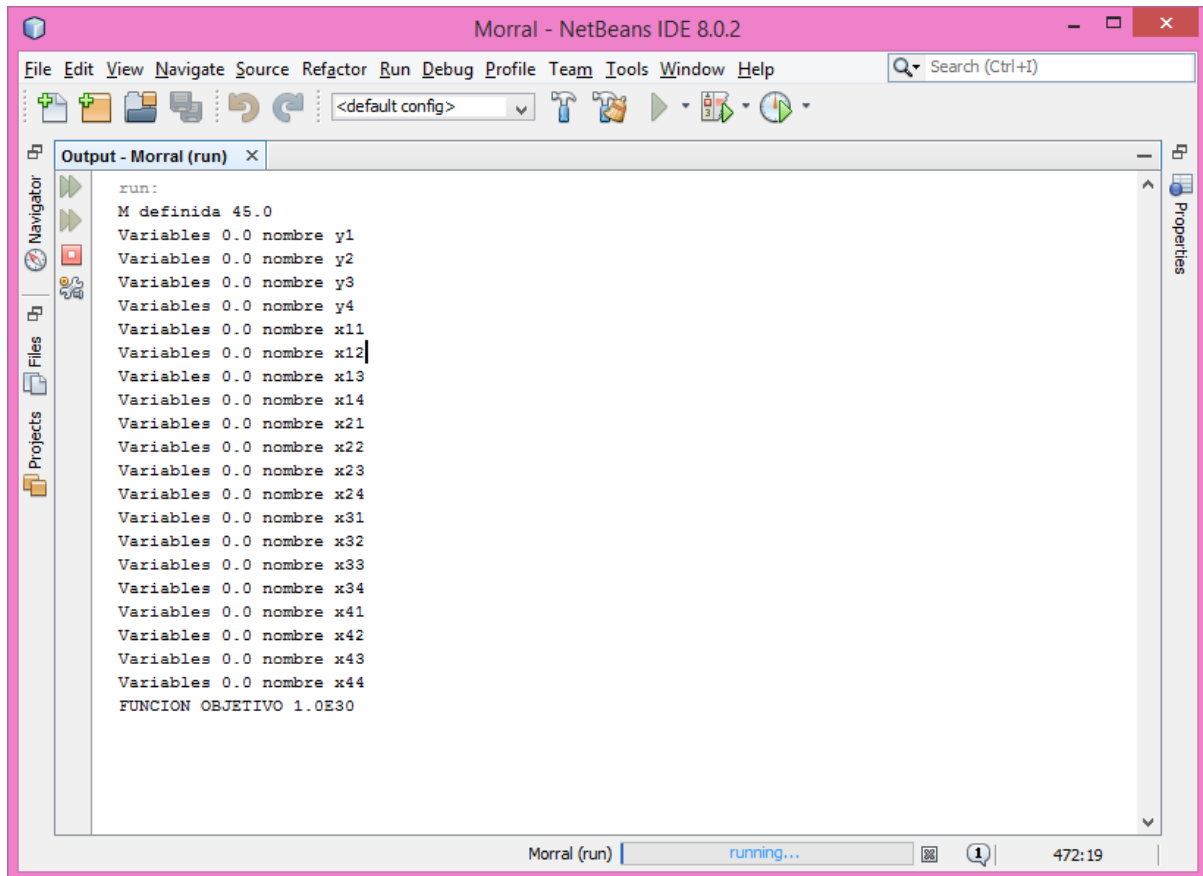
La solucion no es factible

Aceptar

Total minimo de personas  (Total minimo de morrales)

Tiempo de ejecucion  (ms)

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
```

Morral (run) | running... | 472:19

**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**

N°	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

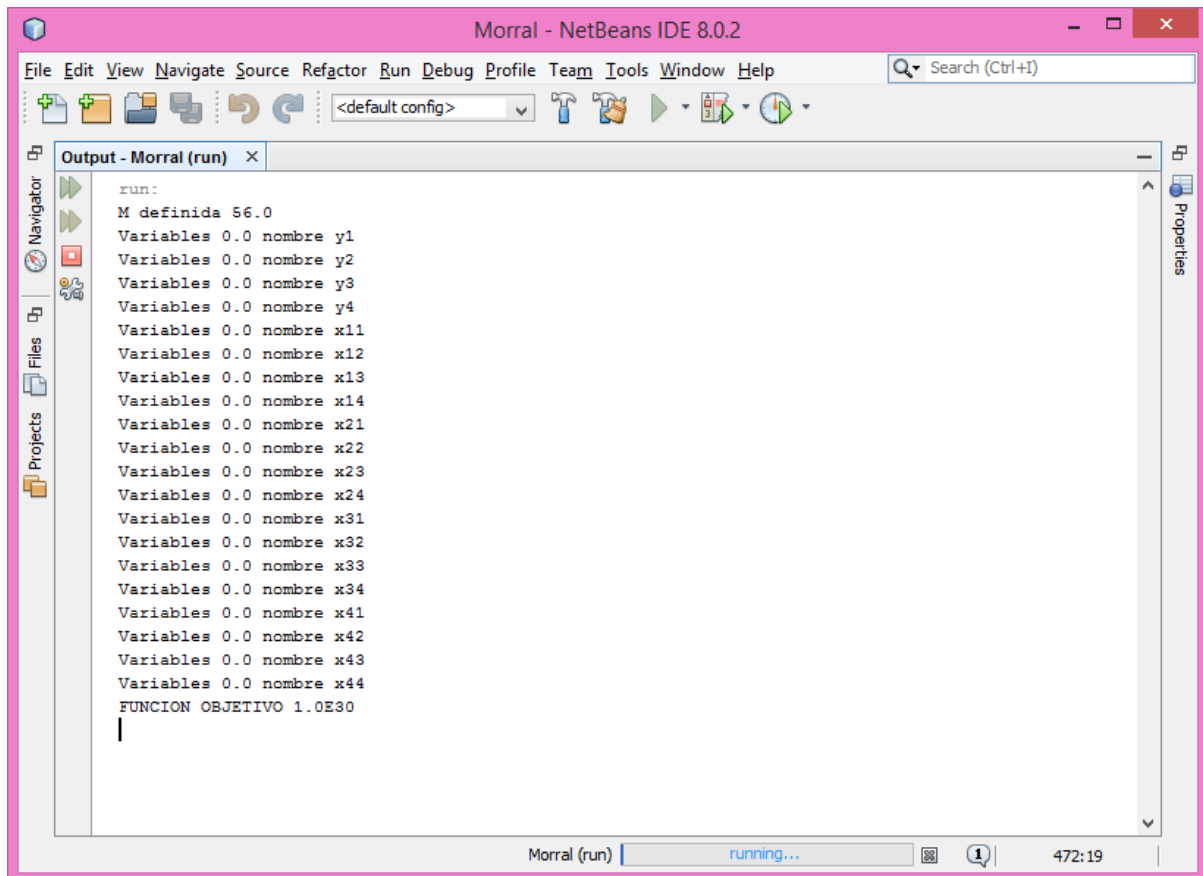
**Total minimo de personas**  (Total minimo de morrales)

**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.



```
run:
M definida 56.0
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
```

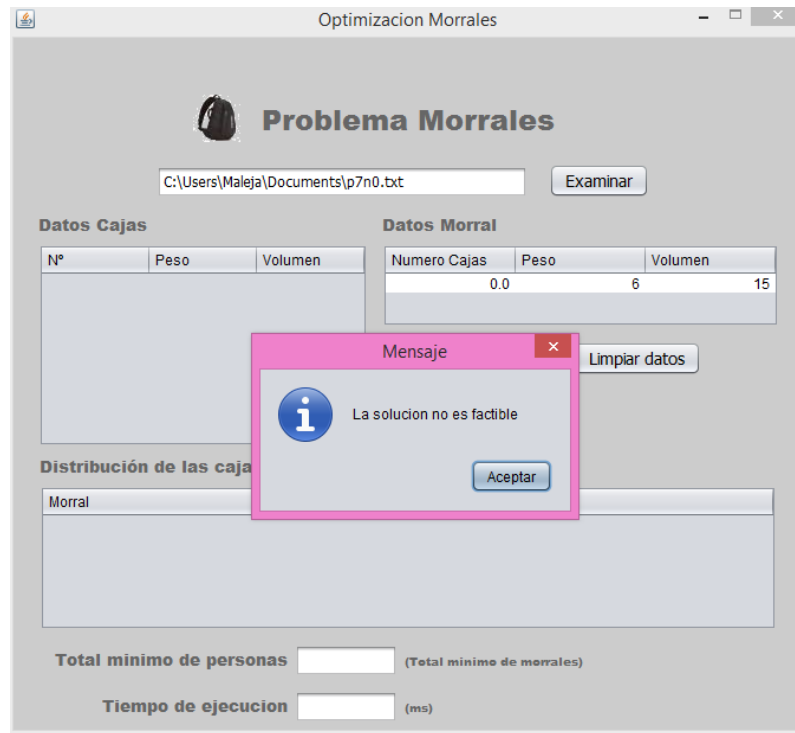
The screenshot shows the NetBeans IDE 8.0.2 interface. The 'Output - Morral (run)' window is active, displaying the output of a Morral run. The output text is as follows:

```
run:
M definida 56.0
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
```

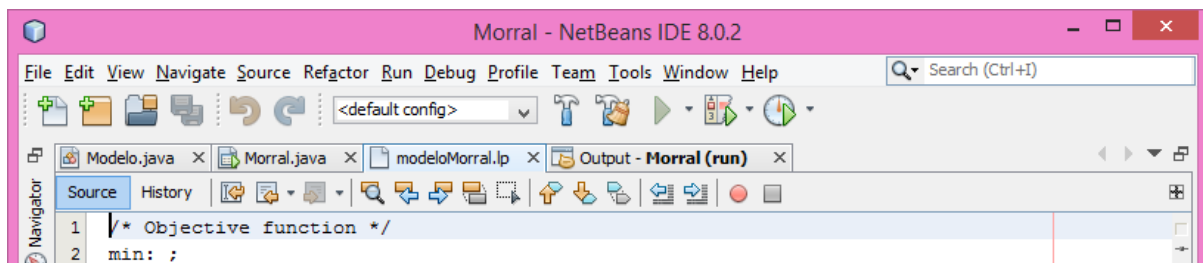
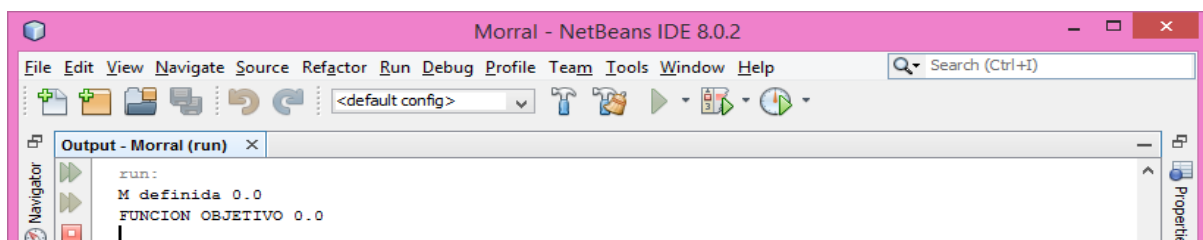
The status bar at the bottom indicates 'Morral (run)' is running, with a progress bar and a timer showing 472:19.

## 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 ejecución 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.

The screenshot shows a software window titled "Optimización Morrales". Inside, there's a section titled "Problema Morrales" with a file path "C:\Users\Maleja\Documents\p9\_Peorcaso.txt" and an "Examinar" button. Below this are two data entry sections: "Datos Cajas" and "Datos Morral".

**Datos Cajas**

N°	Peso	Volumen
1	2	5
2	1	4

**Datos Morral**

Numero Cajas	Peso	Volumen
2.0	2	5

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

**Distribución de las cajas en los morrales**

Morral	Cajas
1	Caja 1;
2	Caja 2;

At the bottom, there are two summary fields: "Total minimo de personas" with a value of 2.0 (Total minimo de morrales) and "Tiempo de ejecucion" with a value of 0.043 (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 12.0
Variables 1.0 nombre y1
Variables 1.0 nombre y2
Variables 1.0 nombre x11
Variables 0.0 nombre x12
Variables 0.0 nombre x21
Variables 1.0 nombre x22
FUNCION OBJETIVO 2.0

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      0.166666666667 after      7 iter is B&B base.

Feasible solution          2 after      12 iter,      3 nodes (gap 50.0%)

Optimal solution          2 after      15 iter,      6 nodes (gap 50.0%) .

Excellent numeric accuracy ||*|| = 0
```

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

Source History

```
1 /* Objective function */
2 min: +y1 +y2;
3
4 /* Constraints */
5 +x11 +x12 = 1;
6 +x21 +x22 = 1;
7 +2 x11 +x21 <= 2;
8 +2 x12 +x22 <= 2;
9 +5 x11 +4 x21 <= 5;
10 +5 x12 +4 x22 <= 5;
11 -12 y1 +x11 +x21 <= 0;
12 -12 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 */
```

Find: variables Previous Next No matches x

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

### 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\_Mejor caso.txt

**Datos Cajas**

Nº	Peso	Volumen
1	2	5
2	3	1

**Datos Morral**

Numero Cajas	Peso	Volumen
2.0	8	9

**Distribución de las cajas en los morrales**

Morral	Cajas
1	Caja 1;Caja 2;

**Total minimo de personas**  (Total minimo de morrales)

**Tiempo de ejecucion**  (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

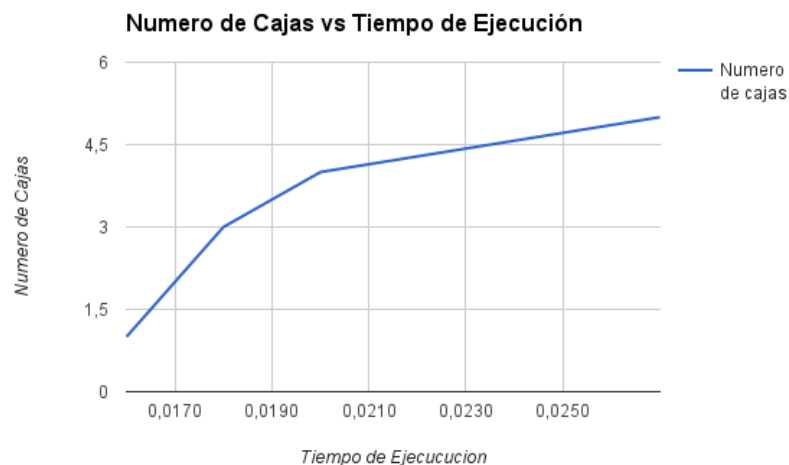
Morral (run) | running... | 1:1 | INS

## 5. ANÁLISIS DE RESULTADOS

Tabla 4. Número de cajas y tiempos de ejecución.

Número de cajas	Tiempo ejecución (ms)
1	0,0160
2	0,0170
3	0,0180
4	0,0200
5	0,0270

Gráfica 1. Número de cajas y tiempos de ejecución.

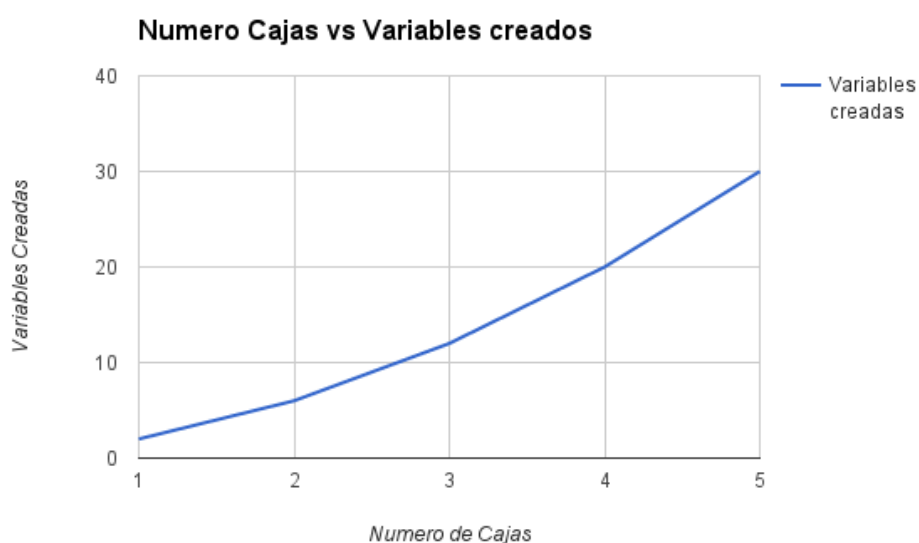


Para realizar el análisis del tiempo de ejecución se debe tener en cuenta el tamaño de las entradas cantidad de cajas y la aproximación de los pesos y volúmenes de cada caja a la capacidad máxima del morral. Según las pruebas realizadas el tiempo de ejecución se ve afectado notablemente con el incremento 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  $v_i$ ). Se puede afirmar, a modo de conclusión, que el tiempo de ejecución se ve afectado por la cantidad total de cajas, manteniendo las características de las cajas y los morrales iguales. El mejor y peor caso si dependen de las características de las cajas si se acercan a la capacidad máxima del morral hacen que cada caja se guarde en un morral.

**Tabla 5. Número de cajas vs cantidad de variables creadas**

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

**Gráfica 2. Número de cajas vs cantidad de variables creadas.**



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 esta dada por la siguiente fórmula: número de cajas \*número de cajas + número de cajas

**Tabla 6. Número de cajas vs cantidad de restricciones creadas**

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

**Gráfica 3. Número de cajas vs cantidad de restricciones creadas**



En el análisis realizado para la mismas características de las cajas y la misma capacidad del morrales se observa que la creación de restricciones tiene un comportamiento creciente que hace que la construcción y procesamiento del modelo sea más demorado.

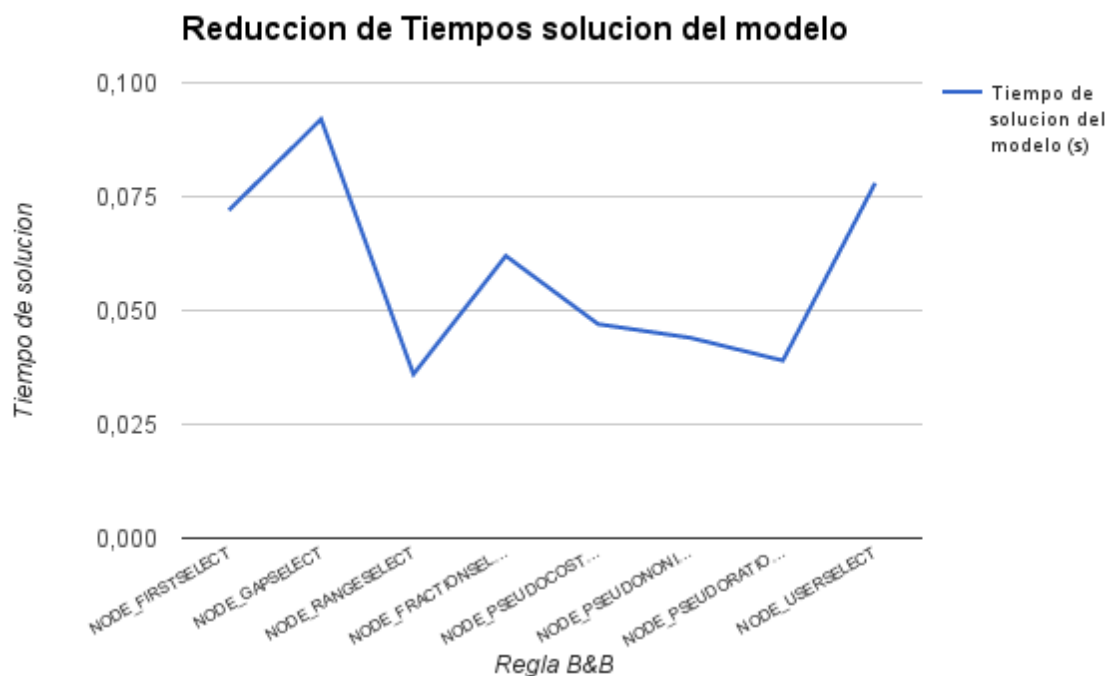
El total de variables creadas esta dada por la siguiente fórmula: número de cajas \*4

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

**Tabla 7. Regla Branch and Bound vs Tiempo de ejecución del modelo**  
Teniendo en cuenta el caso de  $n=20$ .

Regla Branch and Bound	Tiempo de solución del modelo (s)
NODE_FIRSTSELECT	0,072
NODE_GAPSELECT	0,092
NODE_RANGESELECT	0,036
NODE_FRACTIONSELECT	0,062
NODE_PSEUDOCOSTSELECT	0,047
NODE_PSEUDONONINTSELECT	0,044
NODE_PSEUDORATIOSELECT	0,039
NODE_USERSELECT	0,078

**Gráfica 4. Regla Branch and Bound vs Tiempo de ejecución del modelo**



Se probaron varios parámetros para mejorar la forma en como se arma el árbol de “Branch and Bound”, 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\_RANGESELECT, entonces con la instrucción `solver.setBbRule(LpSolve.NODE_RANGESELECT)` 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\_RANGESELECT realiza la selección de acuerdo al “bound” actual. 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.

## 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 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, el objetivo de la segunda parte del proyecto es mejorar esta distribución.

Este problema de morrales se considera como NPC, es decir, un tipo de problema NP difícil. 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.