UNIVERSIDAD DE GRANADA

Simulación de Sistemas

Práctica 2

Alejandro Manzanares Lemus

alexmnzlms@correo.ugr.es

12 de noviembre de 2020

Índice general

Apartado 1:
Mi Segundo Modelo de Simulación de Monte Carlo

1.1: Modelización por montecarlo

1.1.1: Planteamiento del problema

El sistema que vamos a modelizar trata de un quiosco de periódicos que se abastece diariamente de un cierto número de periódicos.

Por cada periódico vendido se obtiene una ganancia de x euros y por cada periódico no vendido se pierde una cantidad de y euros.

El número de periódicos solicitados al proveedor es s.

La demanda D es el número de periódicos que vende el quiosco cada día y obedece a una distribución de probabilidad P(D = d).

A continuación se definen las 3 distribuciones de probabilidad para D.

El objetivo de esta simulación es encontrar el número de periódicos que se solicitan — el valor de s— para el que la ganancia es máxima.

1.1.2: Distribuciones de datos

- Distribución a: $P(D = d) = \frac{1}{100}$, $\forall 0, ..., 99$
  Esta es la distribución uniforme, es decir, todos los valores para D tiene la misma probabilidad.
- Distribución b: $P(D = d) = \frac{100-d}{5050}$, $\forall 0, ..., 99$
  Esta es la distribución proporcional, los valores más pequeños de d tienen mayor probabilidad cuanto más pequeños sean.
- Distribución c: $P(D = d) = \frac{d}{2500}$ si $0 \leq d < 50$ y $P(D = d) = \frac{100-d}{2500}$ si $50 \leq d \leq 99$
  Esta es la distribución triangular, lo que significa que los valores centrales de d tiene mayor probabilidad que los valores extremos, es decir, si $0 \leq d \leq 99$ entonces 0 y 99 tienen una probabilidad muy baja mientras que 50 — que es el valor medio — tiene la probabilidad más alta.

1.1.3: Análisis de resultados

A continuación pueden verse los resultados que se han obtenido para esta simulación variando le valor de y, es decir, modificando cuanto se pierde por cada unidad no vendida. El número de iteraciones ha sido de 100000, lo que quiere decir, que para cada posible valor de s, se han generado 100000 valores aleatorios para D siguiendo las diferentes distribuciones de probabilidad.

Distribución a				
Veces	**Ganancia/Unidad**	**Perdida/Unidad**	**Unidades**	**Ganancia**
100000	10	1	90	449.356
100000	10	5	66	327.772
100000	10	10	50	245.066

Distribución b				
Veces	**Ganancia/Unidad**	**Perdida/Unidad**	**Unidades**	**Ganancia**
100000	10	1	69	283.013
100000	10	5	42	187.965
100000	10	10	29	133.422

Distribución c				
Veces	**Ganancia/Unidad**	**Perdida/Unidad**	**Unidades**	**Ganancia**
100000	10	1	77	464.006
100000	10	5	59	386.296
100000	10	10	50	333.032

Para la distribución a:

Podemos observar como, al aumentar el valor de pérdida/unidad, el valor óptimo de s disminuye, es decir, cuanto más aumenta lo que perdemos por una unidad, menos rentable resulta pedir muchas unidades, porque al ser la demanda igualmente probable, es igualmente posible vender 0 unidades que vender 99.

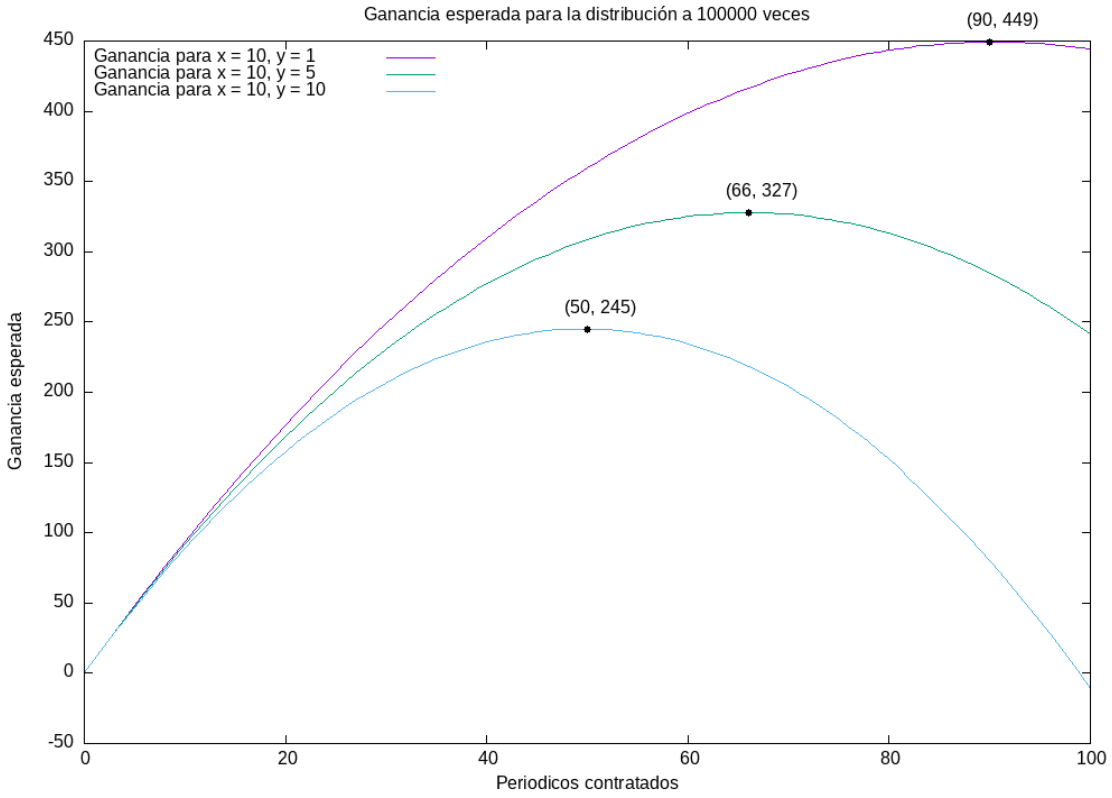
Para la distribución b:

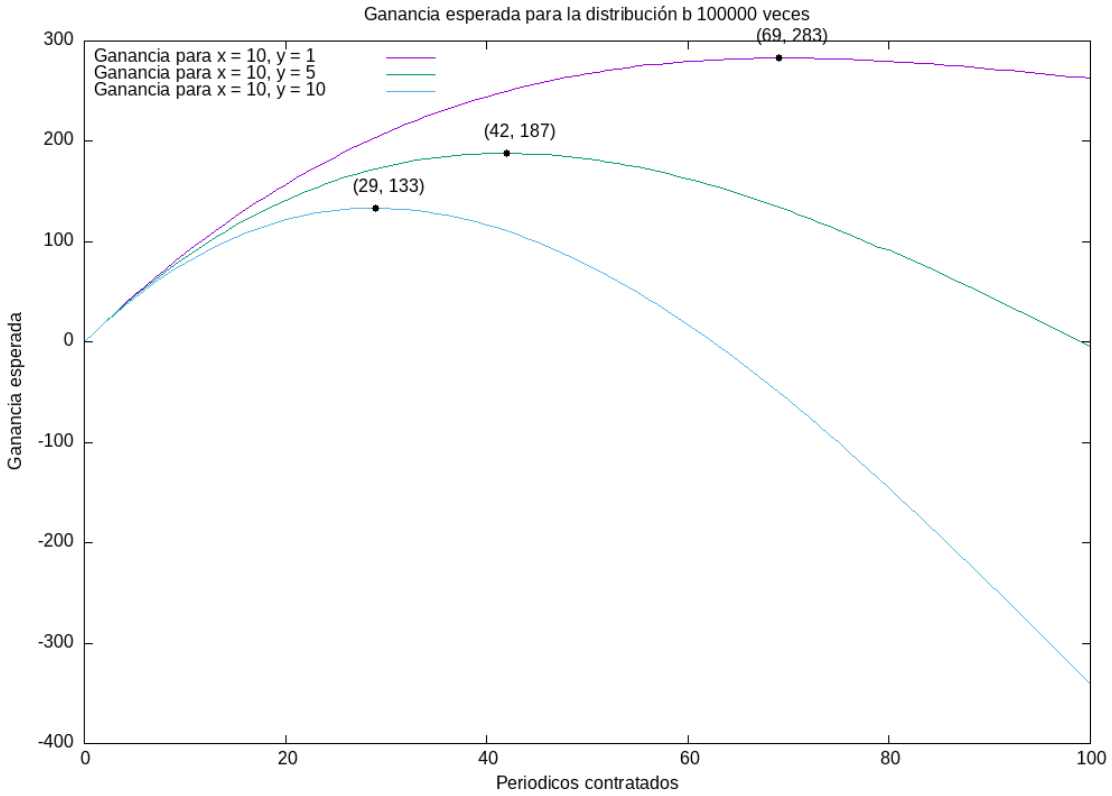
En este caso podemos ver como, al ser más posible vender pocas unidades que vender mucho, es muy poco rentable encargar muchos periódicos cuanto mayor es la pérdida que obtenemos al dejar periódicos sin vender. Por el mismo motivo, las ganancias máximás se reducen, puesto que lo más probable es vender pocos periódicos.

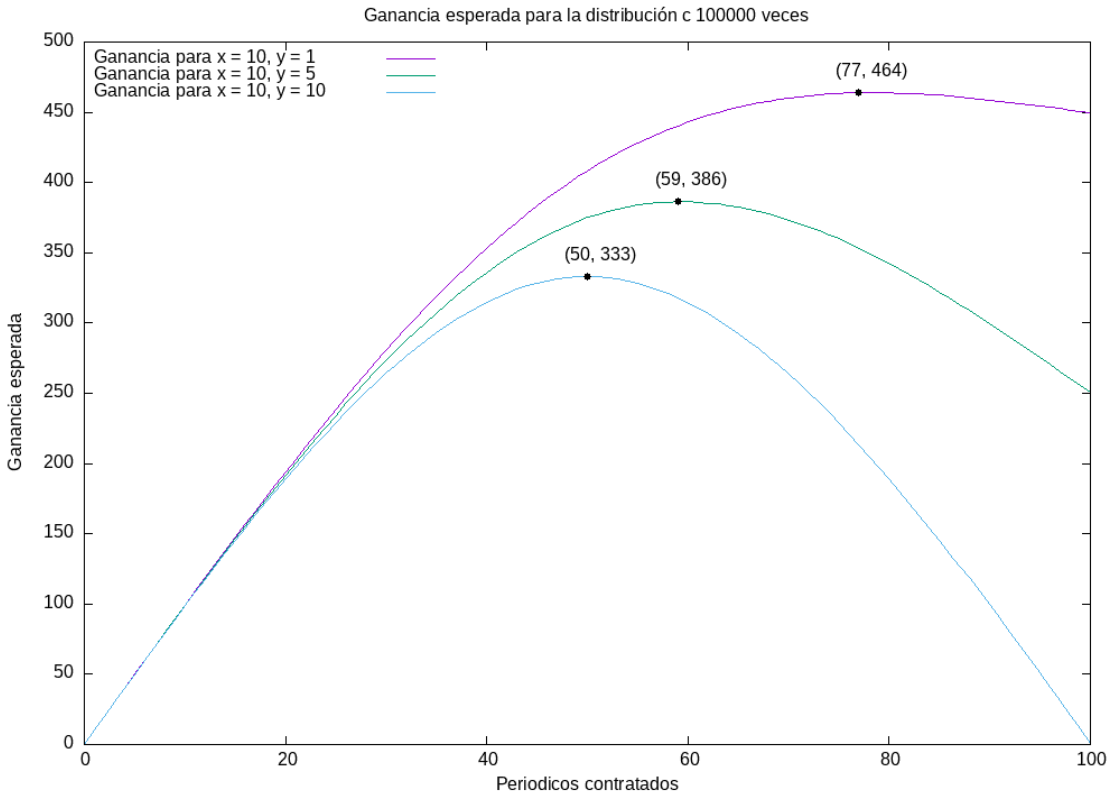
Para la distribución c:

Vemos que es bastante parecida a los resultados obtenidos para la distribución a. Esto tiene sentido puesto que la probabilidad de que salgan valores intermedios es mayor, por lo que de media obtendremos datos bastante parecidos.

A continuación se pueden apreciar las gráficas de la ganancia obtenida para cada valor de s.







1.1.4: Tablas de datos

A continuación se muestran los mismos datos, pero con variaciones en el número de ejecuciones que se realizan para cada valor de s.

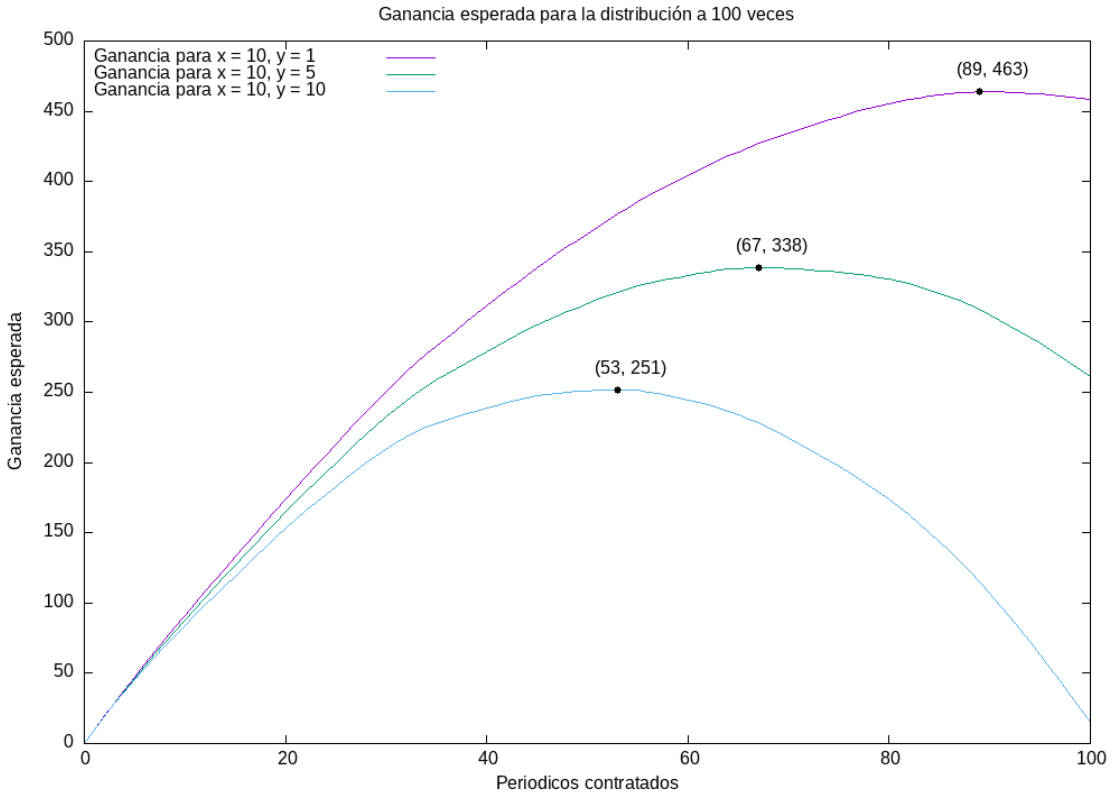
Distribución a				
Veces	**Ganancia/Unidad**	**Perdida/Unidad**	**Unidades**	**Ganancia**
100	10	1	89	463.86
100	10	5	67	338.65
100	10	10	53	251.8
1000	10	1	90	453.356
1000	10	5	68	329.135
1000	10	10	51	241.68
5000	10	1	91	454.153
5000	10	5	68	330.752
5000	10	10	49	239.772
10000	10	1	90	451.672
10000	10	5	67	330.668
10000	10	10	50	246.502
100000	10	1	90	449.356
100000	10	5	66	327.772
100000	10	10	50	245.066

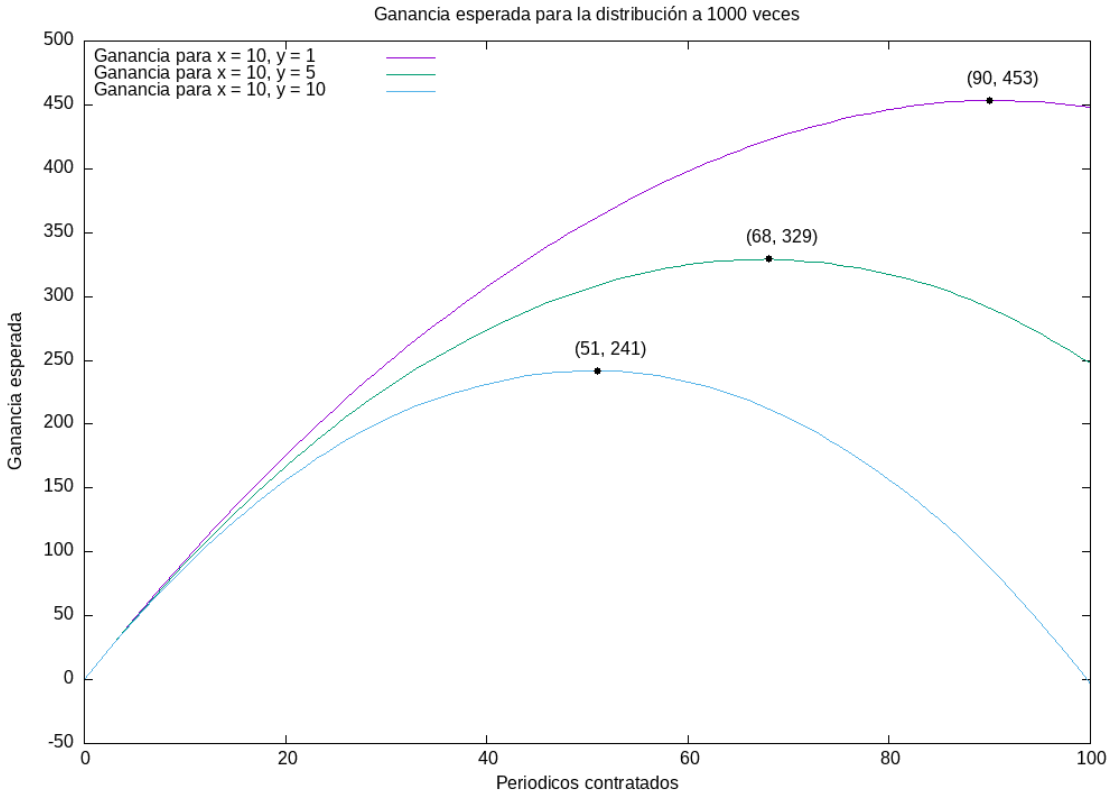
Distribución b				
Veces	**Ganancia/Unidad**	**Perdida/Unidad**	**Unidades**	**Ganancia**
100	10	1	68	296.65
100	10	5	43	196.3
100	10	10	31	138.4
1000	10	1	70	287.731
1000	10	5	43	189.82
1000	10	10	30	132.2
5000	10	1	71	287.868
5000	10	5	43	190.486
5000	10	10	29	130.584
10000	10	1	70	284.796
10000	10	5	43	190.042
10000	10	10	29	134.678
100000	10	1	69	283.013
100000	10	5	42	187.965
100000	10	10	29	133.422

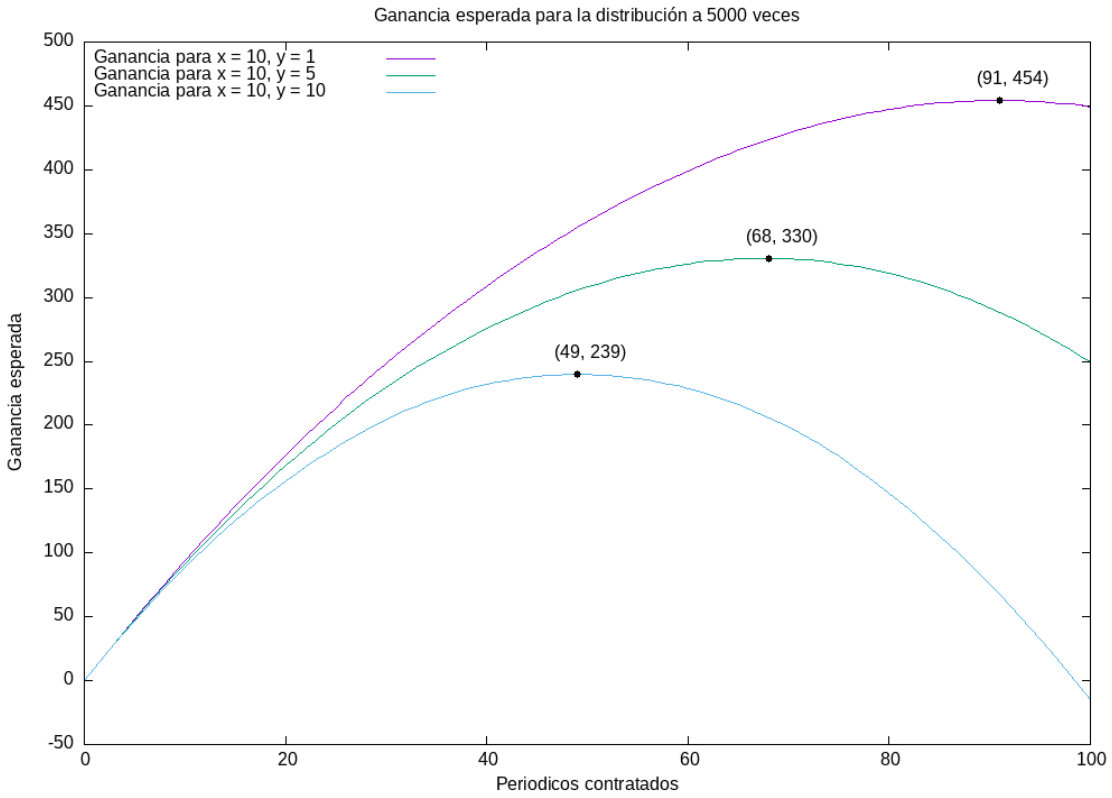
Distribución c				
Veces	**Ganancia/Unidad**	**Perdida/Unidad**	**Unidades**	**Ganancia**
100	10	1	77	472.23
100	10	5	60	389.85
100	10	10	52	334.4
1000	10	1	79	466.281
1000	10	5	60	385.515
1000	10	10	51	330.16
5000	10	1	79	467.379
5000	10	5	59	383.501
5000	10	10	50	329.704
10000	10	1	79	465.595
10000	10	5	60	387.565
10000	10	10	50	334.46
100000	10	1	77	464.006
100000	10	5	59	386.296
100000	10	10	50	333.032

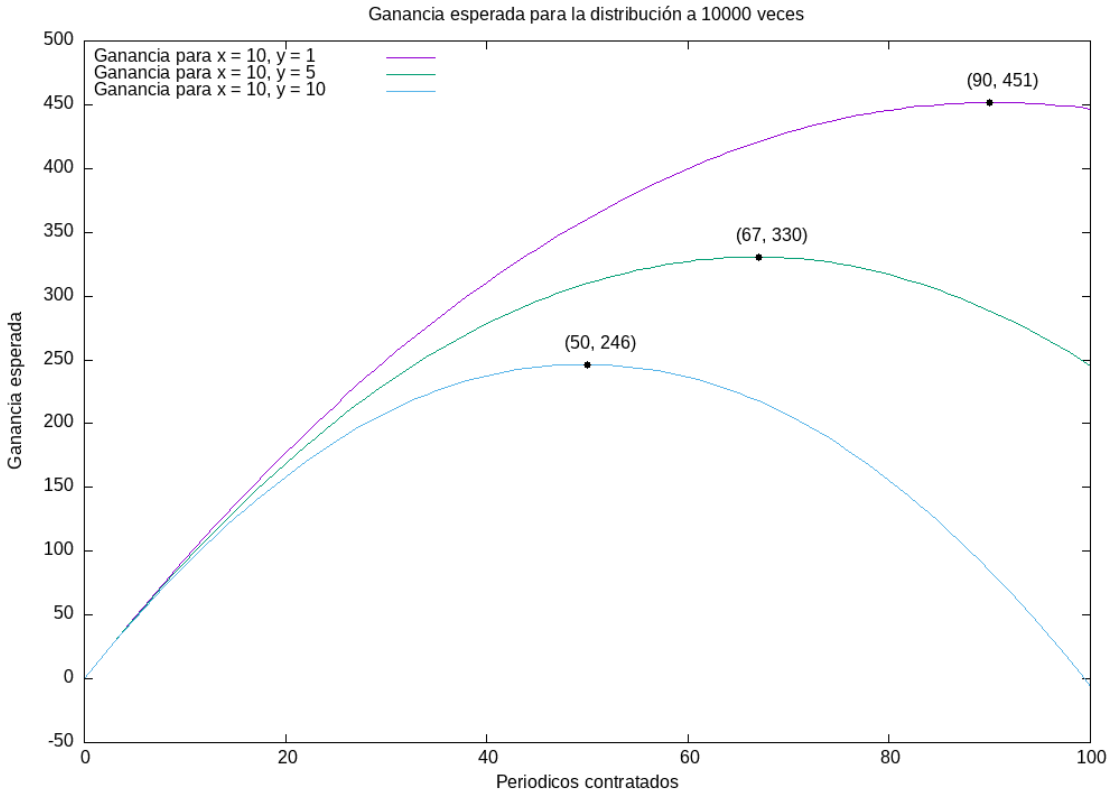
1.1.5: Gráficas

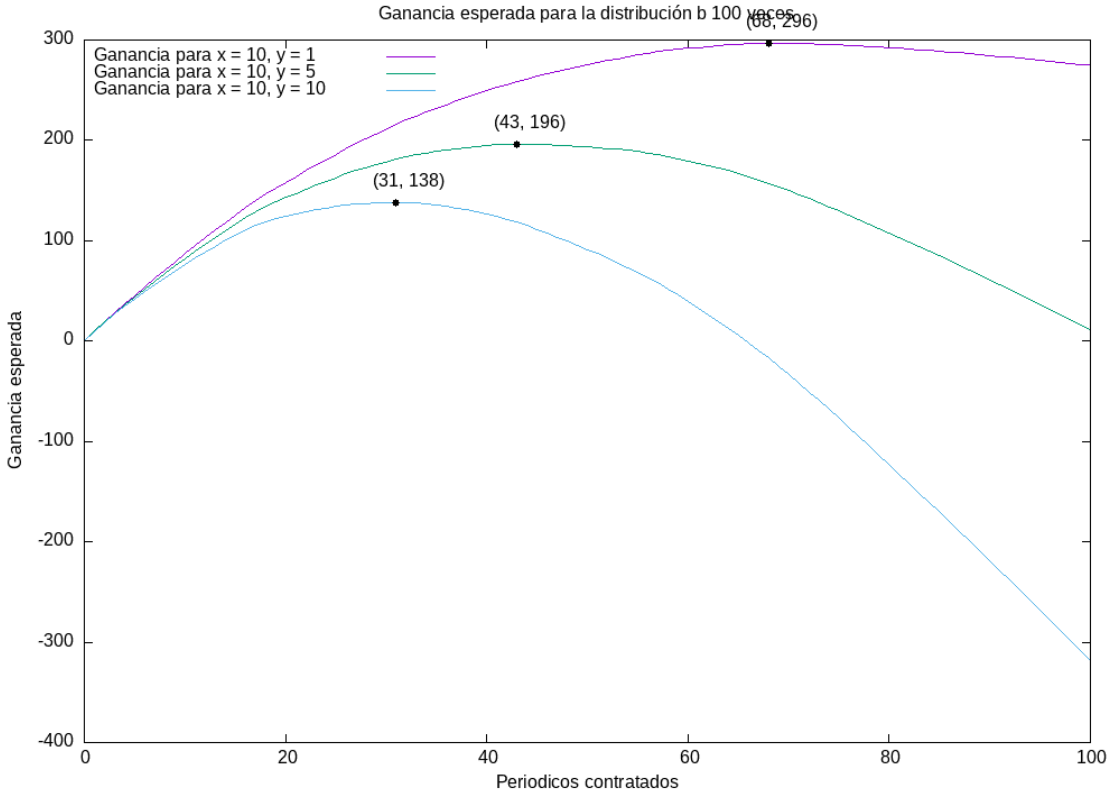
Aquí podemos ver los datos de manera gráfica.

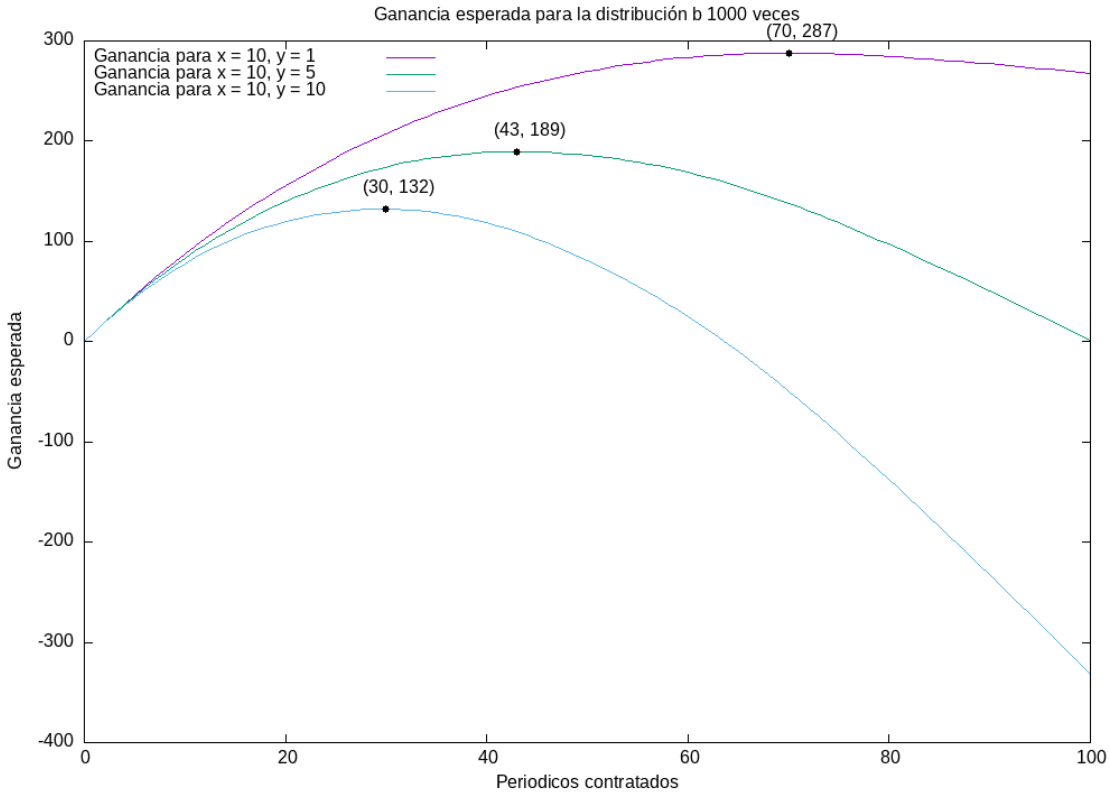


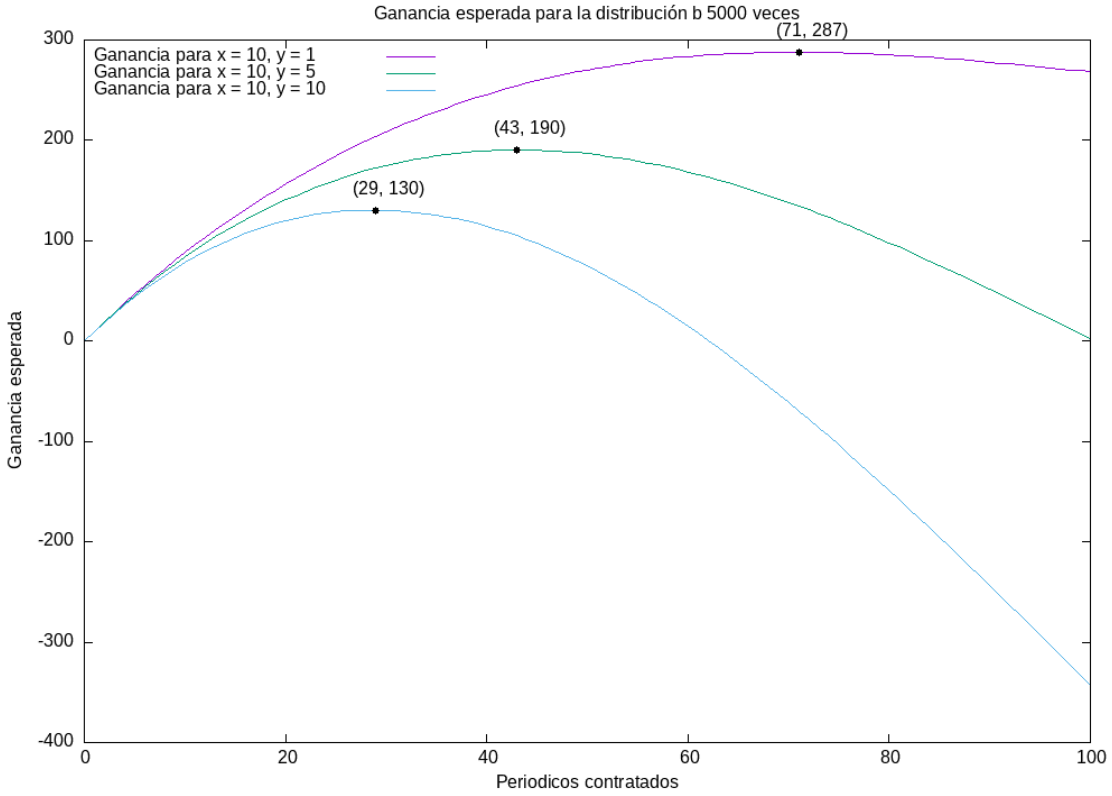


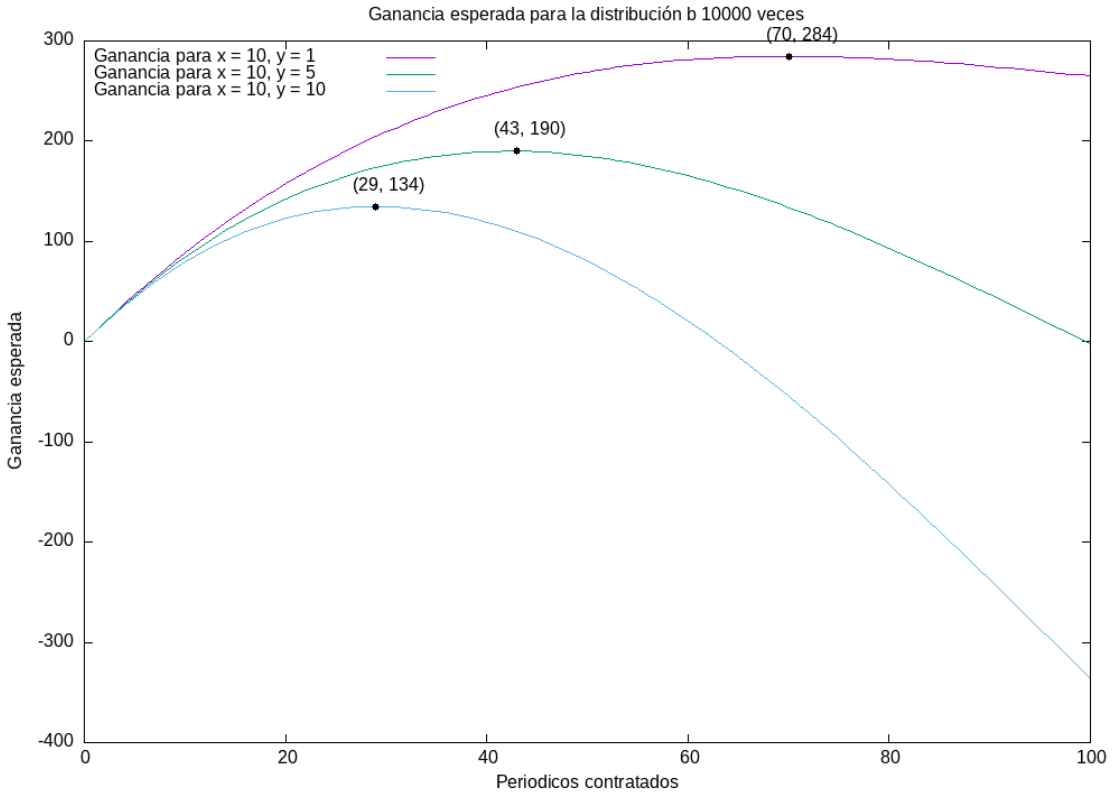

Ganancia esperada para la distribución a 5000 veces
Ganancia para x = 10, y = 1
Ganancia para x = 10, y = 5
Ganancia para x = 10, y = 10
(91, 454)
(68, 330)
(49, 239)
Ganancia esperada
Periodicos contratados
500
450
400
350
300
250
200
150
100
50
0
-50
0
20
40
60
80
100

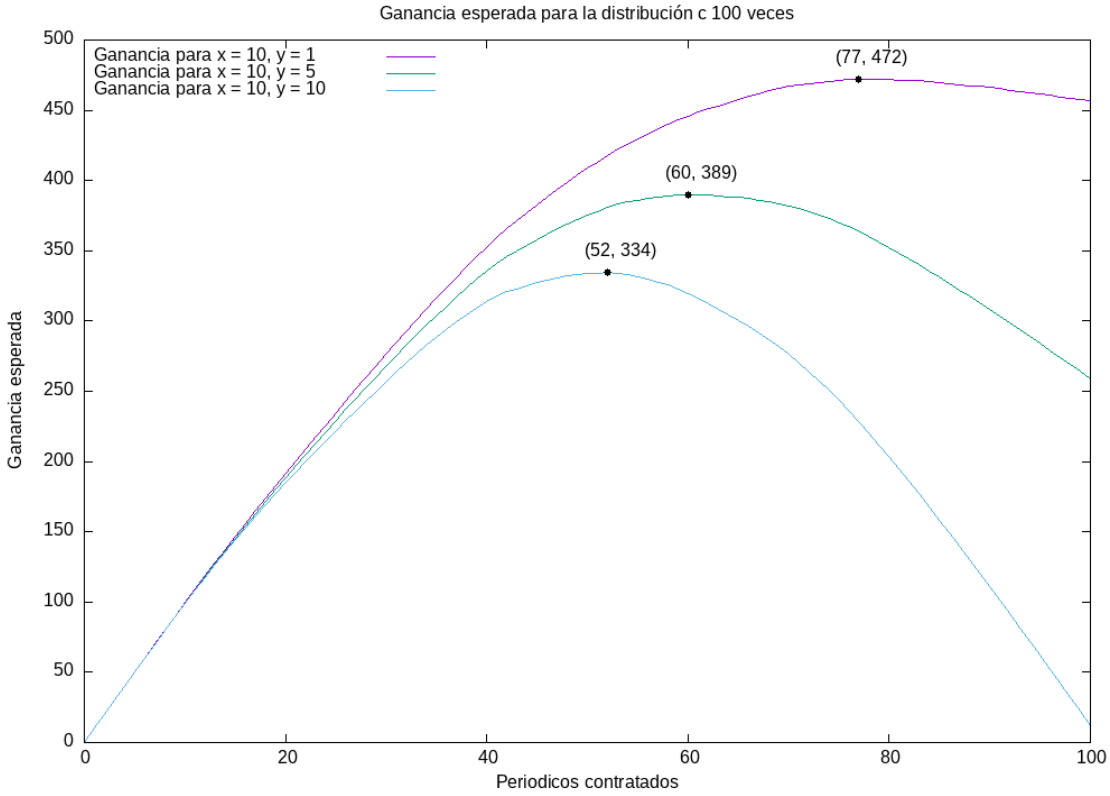

Ganancia esperada para la distribución a 10000 veces
Ganancia para x = 10, y = 1
Ganancia para x = 10, y = 5
Ganancia para x = 10, y = 10
(90, 451)
(67, 330)
(50, 246)
Ganancia esperada
Periodicos contratados
500
450
400
350
300
250
200
150
100
50
0
-50
0
20
40
60
80
100

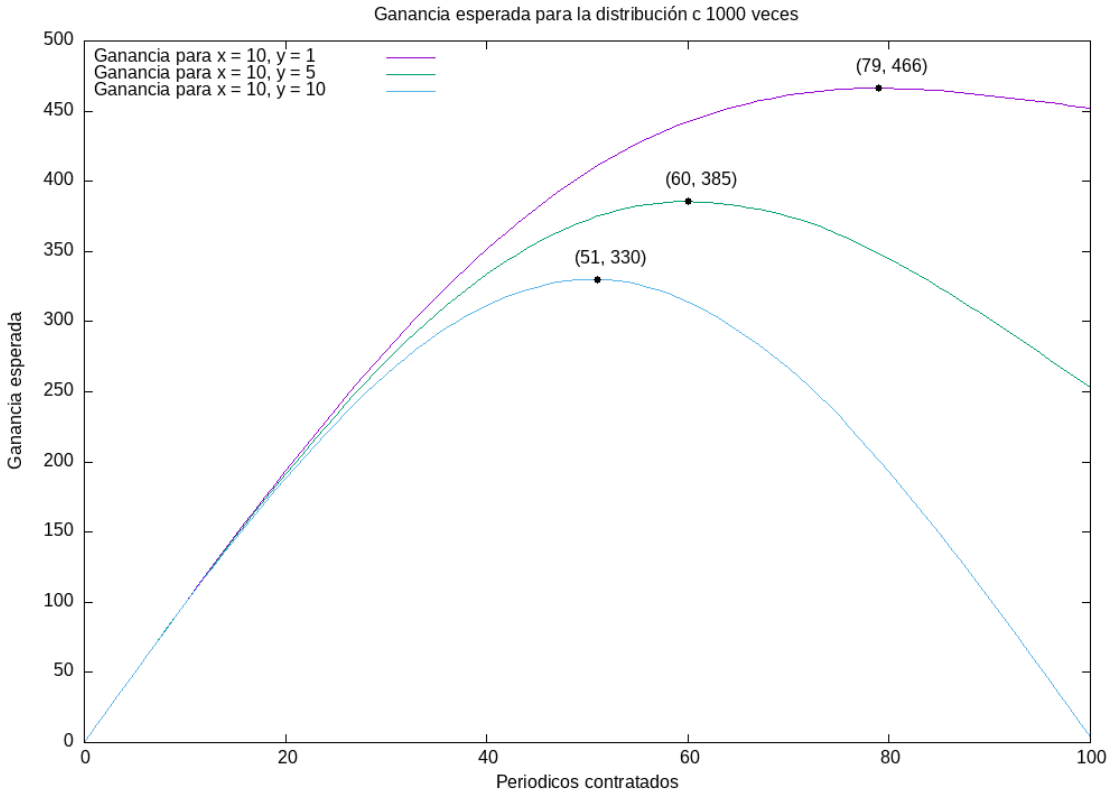

Ganancia esperada para la distribución b 100 veces
Ganancia para x = 10, y = 1
Ganancia para x = 10, y = 5
Ganancia para x = 10, y = 10
(68, 296)
(43, 196)
(31, 138)
Ganancia esperada
Periodicos contratados
300
200
100
0
-100
-200
-300
-400
0
20
40
60
80
100

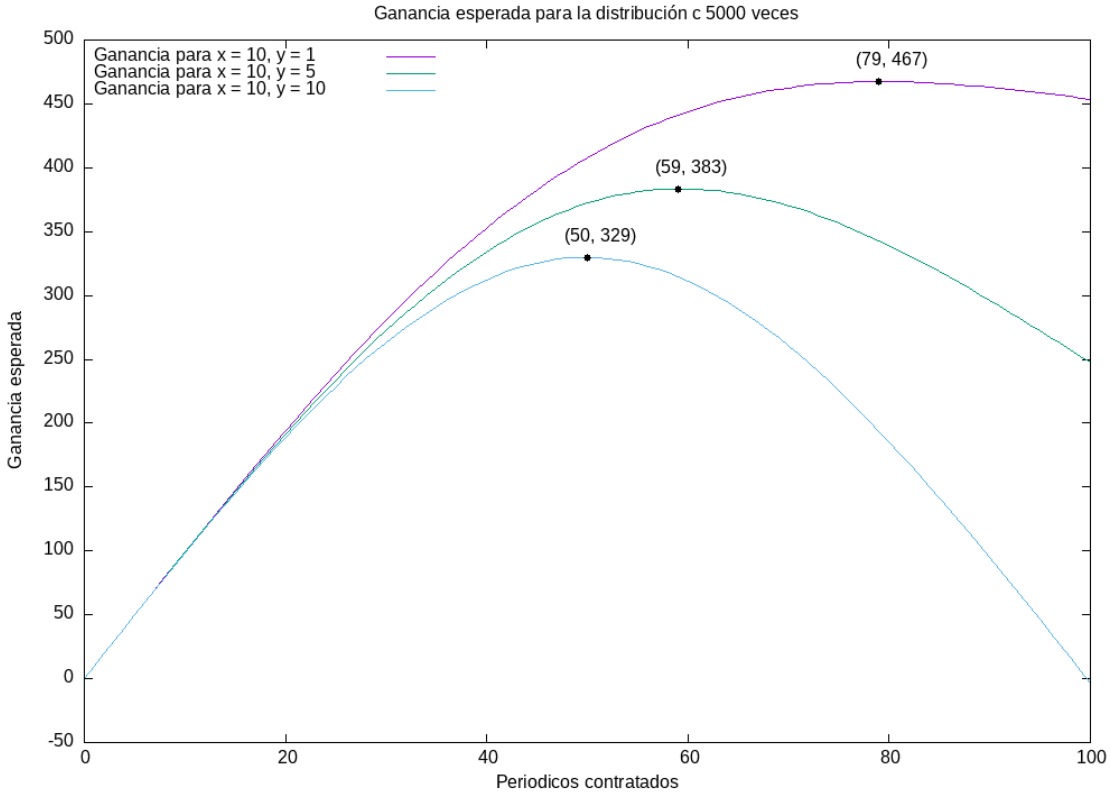

Ganancia esperada para la distribución b 1000 veces
Ganancia para x = 10, y = 1
Ganancia para x = 10, y = 5
Ganancia para x = 10, y = 10
(70, 287)
(43, 189)
(30, 132)
Ganancia esperada
Periodicos contratados
300
200
100
0
-100
-200
-300
-400
0
20
40
60
80
100

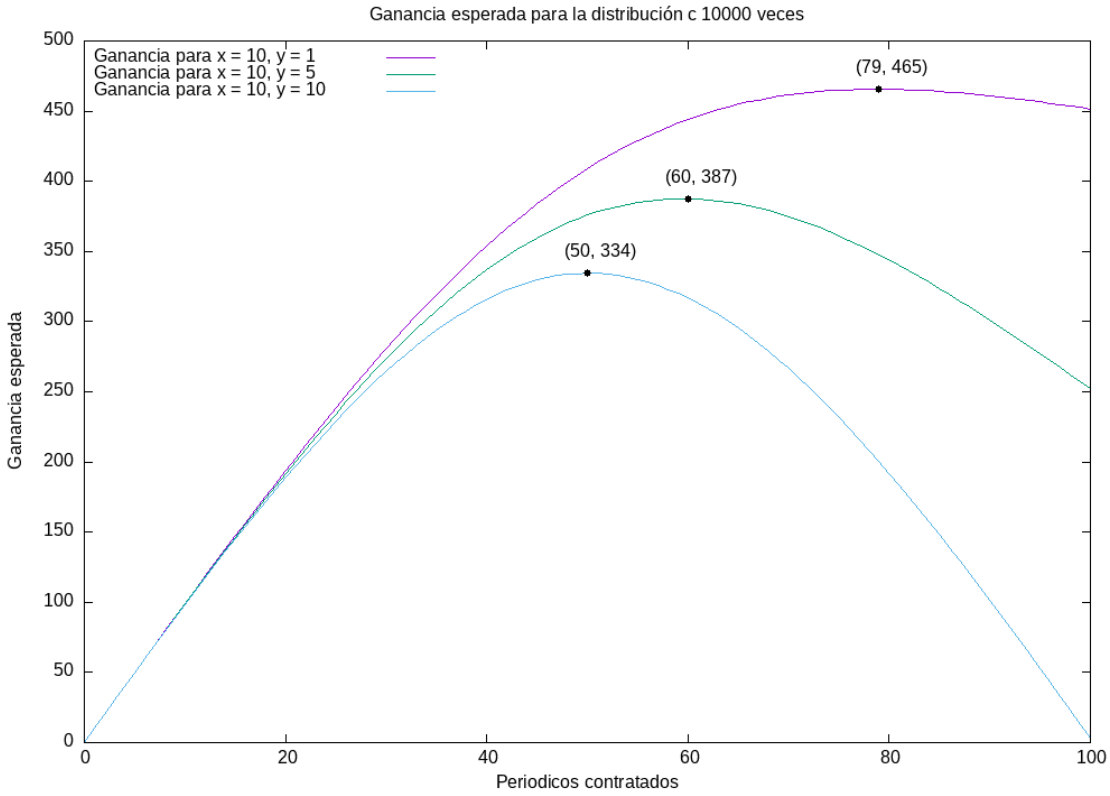

Ganancia esperada para la distribución b 5000 veces
Ganancia para x = 10, y = 1
Ganancia para x = 10, y = 5
Ganancia para x = 10, y = 10
(71, 287)
(43, 190)
(29, 130)
Ganancia esperada
Periodicos contratados
300
200
100
0
-100
-200
-300
-400
0
20
40
60
80
100


Ganancia esperada para la distribución b 10000 veces
Ganancia para x = 10, y = 1
Ganancia para x = 10, y = 5
Ganancia para x = 10, y = 10
(70, 284)
(43, 190)
(29, 134)
Ganancia esperada
Periodicos contratados
300
200
100
0
-100
-200
-300
-400
0
20
40
60
80
100


Ganancia esperada para la distribución c 100 veces
Ganancia para x = 10, y = 1
Ganancia para x = 10, y = 5
Ganancia para x = 10, y = 10
(77, 472)
(60, 389)
(52, 334)
Ganancia esperada
Periodicos contratados
0
50
100
150
200
250
300
350
400
450
500
0
20
40
60
80
100


Ganancia esperada para la distribución c 1000 veces
Ganancia para x = 10, y = 1
Ganancia para x = 10, y = 5
Ganancia para x = 10, y = 10
(79, 466)
(60, 385)
(51, 330)
Ganancia esperada
Periodicos contratados
0
50
100
150
200
250
300
350
400
450
500
0
20
40
60
80
100


Ganancia esperada para la distribución c 5000 veces
Ganancia para x = 10, y = 1
Ganancia para x = 10, y = 5
Ganancia para x = 10, y = 10
(79, 467)
(59, 383)
(50, 329)
Ganancia esperada
Periodicos contratados
500
450
400
350
300
250
200
150
100
50
0
-50
0
20
40
60
80
100


Ganancia esperada para la distribución c 10000 veces
Ganancia para x = 10, y = 1
Ganancia para x = 10, y = 5
Ganancia para x = 10, y = 10
(79, 465)
(60, 387)
(50, 334)
Ganancia esperada
Periodicos contratados
500
450
400
350
300
250
200
150
100
50
0
0
20
40
60
80
100

1.2: Modificaciones del modelo

1.2.1: Primera modificación

La primera modificación al sistema consiste en eliminar las pérdidas por unidad no vendida, es decir, y = 0. Sin embargo introducimos un nuevo parámetro z que representa los gastos de devolución de las unidades no vendidas. Es decir, ahora las pérdidas son independientes de cuanto no vendamos, si no que dependen de cuanto cueste devolver las unidades no vendidas.

Podemos ver a continuación los resultados obtenidos en la simulación para esta modificación. Lo primero debe ser notar que los posibles valores de z con los que trabajamos son 100, 500 y 1000. Esto es respectivamente que los costes de devolución equivalen a 10 ventas — porque por cada venta se obtiene un beneficio de 10 —, 50 ventas y 100 ventas, es decir, un 10 % de las ventas, la mitad de las ventas y el total de las ventas, ya que la demanda máxima puede ser 100 periódicos.

Distribución a			
Ganancia/Unidad	**Gastos Devolución**	**Unidades**	**Ganancia**
10	100	90	399.802
10	500	49	127.062
10	1000	0	-0.302

Distribución b			
Ganancia/Unidad	**Gastos Devolución**	**Unidades**	**Ganancia**
10	100	80	235.934
10	500	0	-0.251
10	1000	0	-10.301

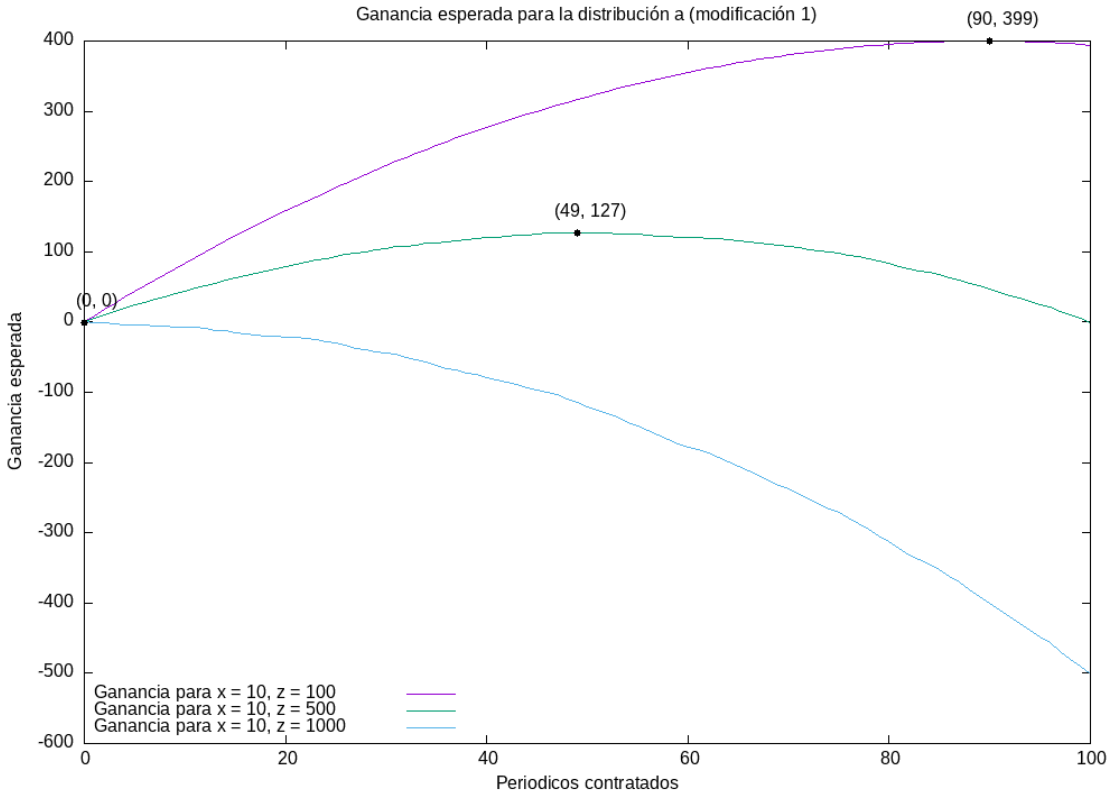
Distribución c			
Ganancia/Unidad	**Gastos Devolución**	**Unidades**	**Ganancia**
10	100	79	405.927
10	500	36	204.343
10	1000	24	119.323

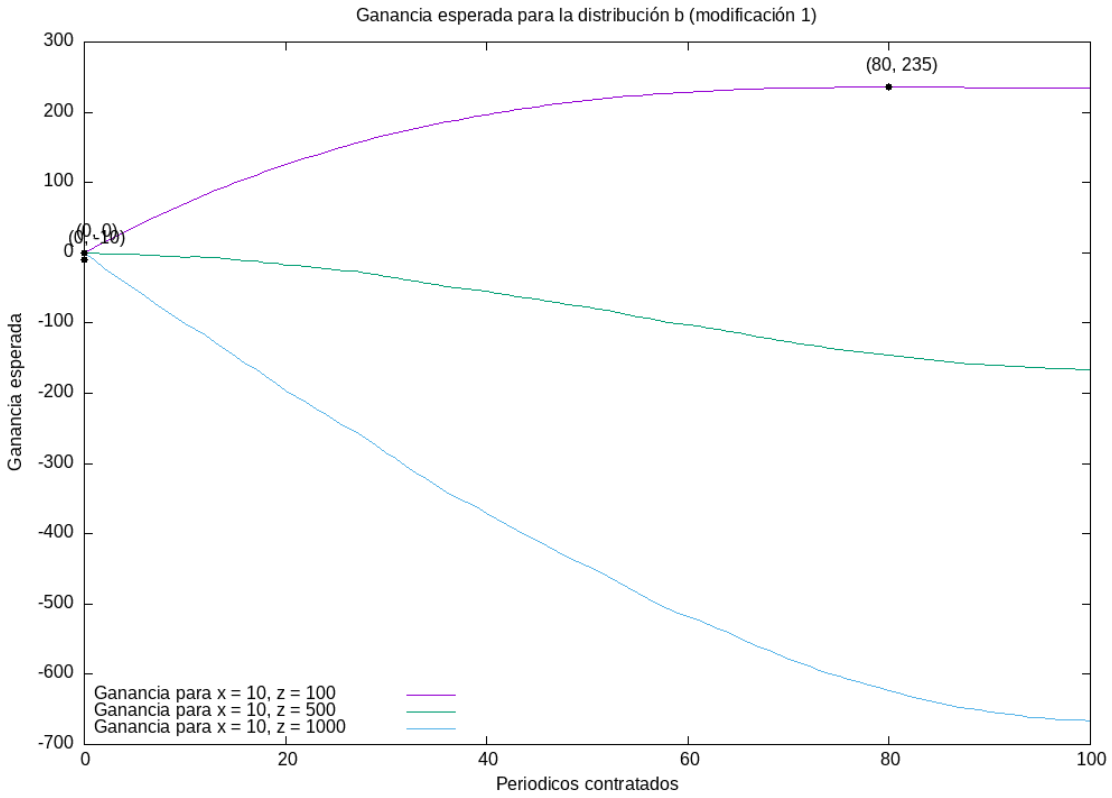
Podemos ver que para unos gastos de devolución de 100€ la ganancia aumenta según aumenta el número de periódicos que contratamos, porque a mayor número de periódicos tengamos, más probable es que vendamos mucho —sobre todo para la distribución a y c — y mayor beneficio obtendremos.

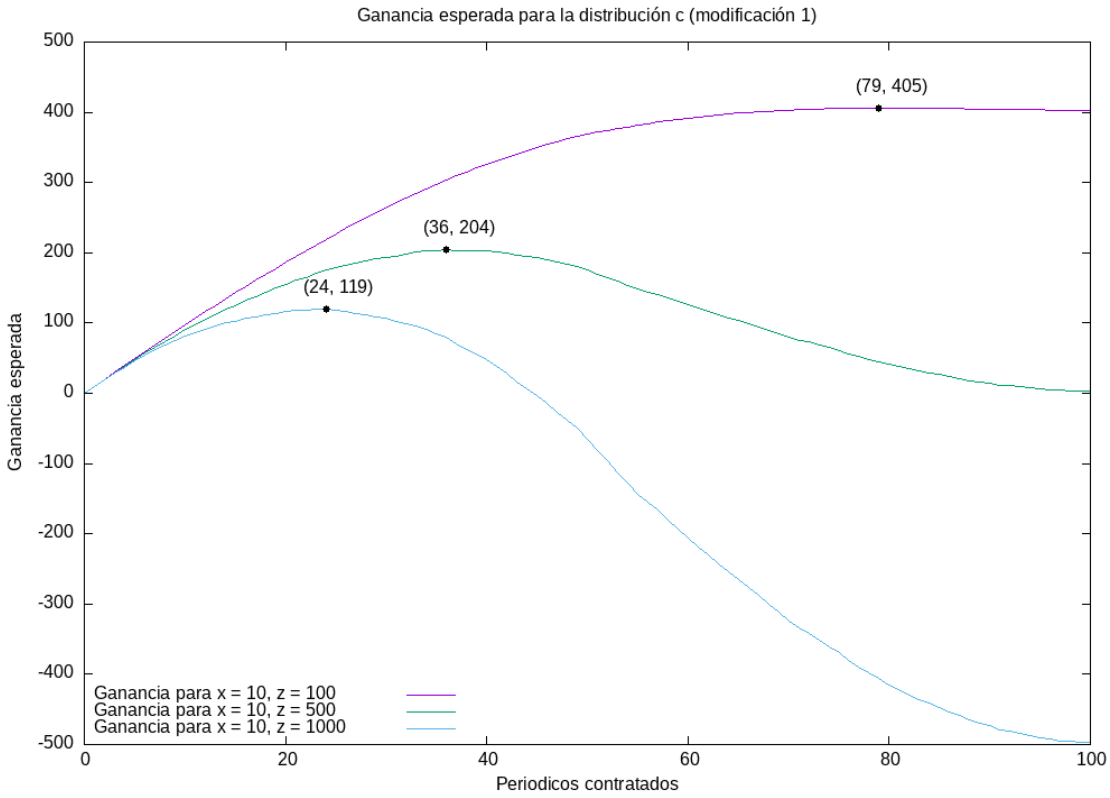
Para los gastos de devolución de 500€ podemos observar como el número de unidades de periódicos necesarias para alcanzar el mismo se reduce drasticamente, llegando a incluso a obtener pérdidas en vez de beneficios. Sobre todo para la distribución de probabilidad b, ya que es mucho menos probable vender un gran número de periódicos.

Para los gastos de devolución de 1000€ vemos que es una opción inviable para las distribuciones a y b, sin embargo, si es una opción posible si trabajamos con la distribución c, aunque los beneficios totales son bastante reducidos.

A continuación podemos ver los resultados obtenidos gráficamente.







1.2.2: Segunda modificación

Finalmente la segunda modificación se basa en una combinación del problema original y la primera modificación. Ahora es posible elegir entre la opción que genere la mínima pérdida de dinero, es decir, podemos escoger entre la opción más rentable entre pagar por cada unidad no vendida — valor de y — y pagar un precio fijo por la devolución del total de unidades no vendidas —valor de z —. En este caso los valores de z e y son los mismos que para los casos anteriores.

Distribución a				
Ganancia/Unidad	**Perdida/Unidad**	**Gastos Devolución**	**Unidades**	**Ganancia**
10	1	100	90	448.202
10	5	100	89	406.718
10	10	100	89	401.75
10	1	500	90	446.33
10	5	500	66	325.459
10	10	500	49	241.906
10	1	1000	90	446.33
10	5	1000	66	325.459
10	10	1000	49	241.906

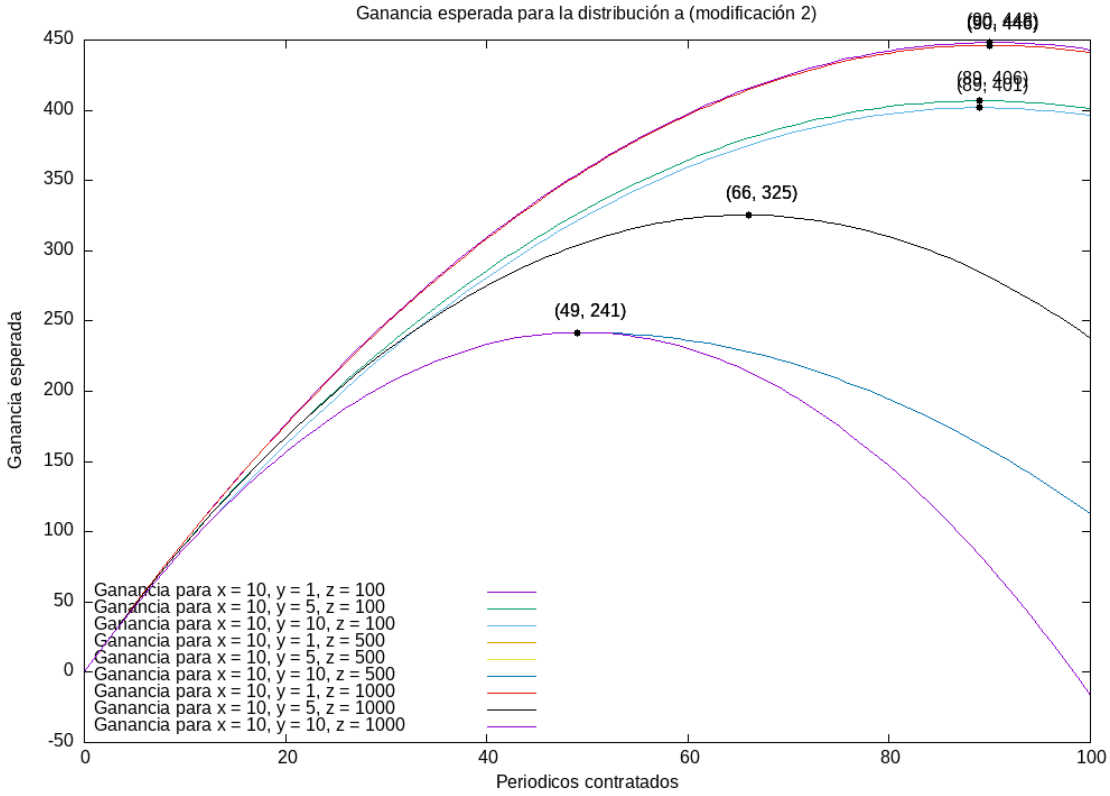
Distribución b				
Ganancia/Unidad	**Perdida/Unidad**	**Gastos Devolución**	**Unidades**	**Ganancia**
10	1	100	69	280.685
10	5	100	71	234.661
10	10	100	75	230.966
10	1	500	69	280.685
10	5	500	42	186.603
10	10	500	29	131.972
10	1	1000	69	280.685
10	5	1000	42	186.603
10	10	1000	29	131.972

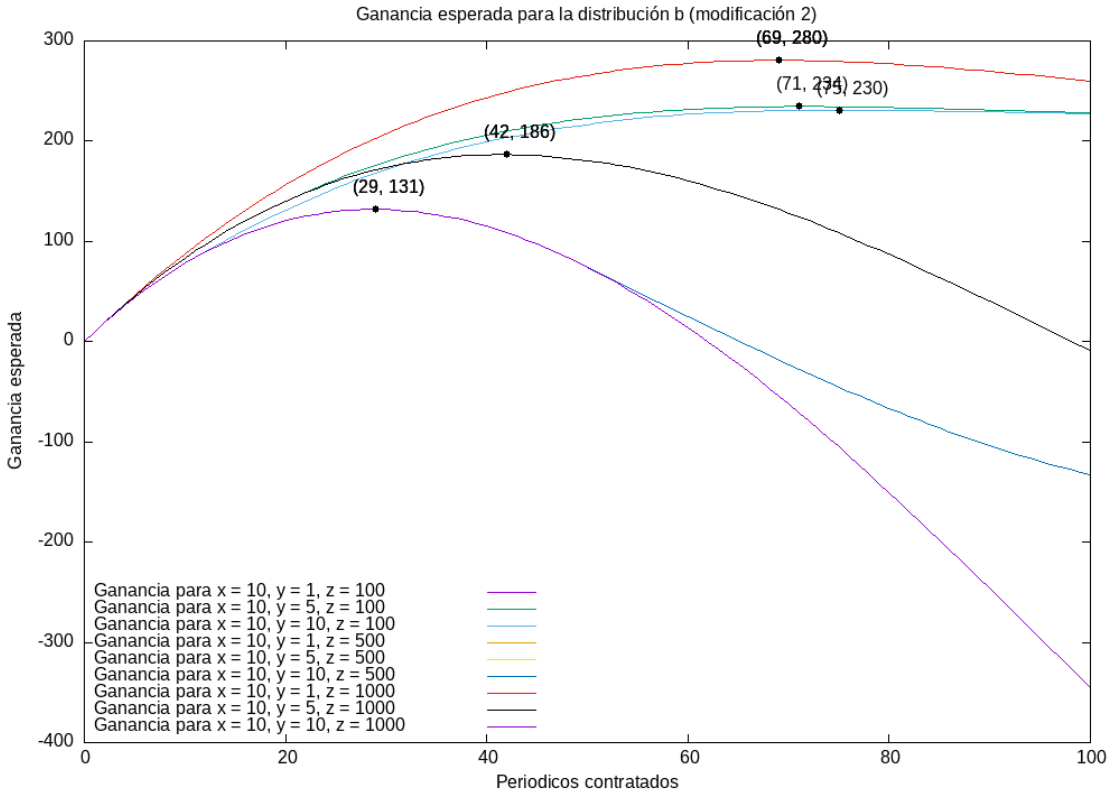
Distribución c				
Ganancia/Unidad	**Perdida/Unidad**	**Gastos Devolución**	**Unidades**	**Ganancia**
10	1	100	78	461.712
10	5	100	72	412.534
10	10	100	75	405.259
10	1	500	78	461.712
10	5	500	59	383.876
10	10	500	50	329.892
10	1	1000	79	461.563
10	5	1000	59	382.647
10	10	1000	50	329.892

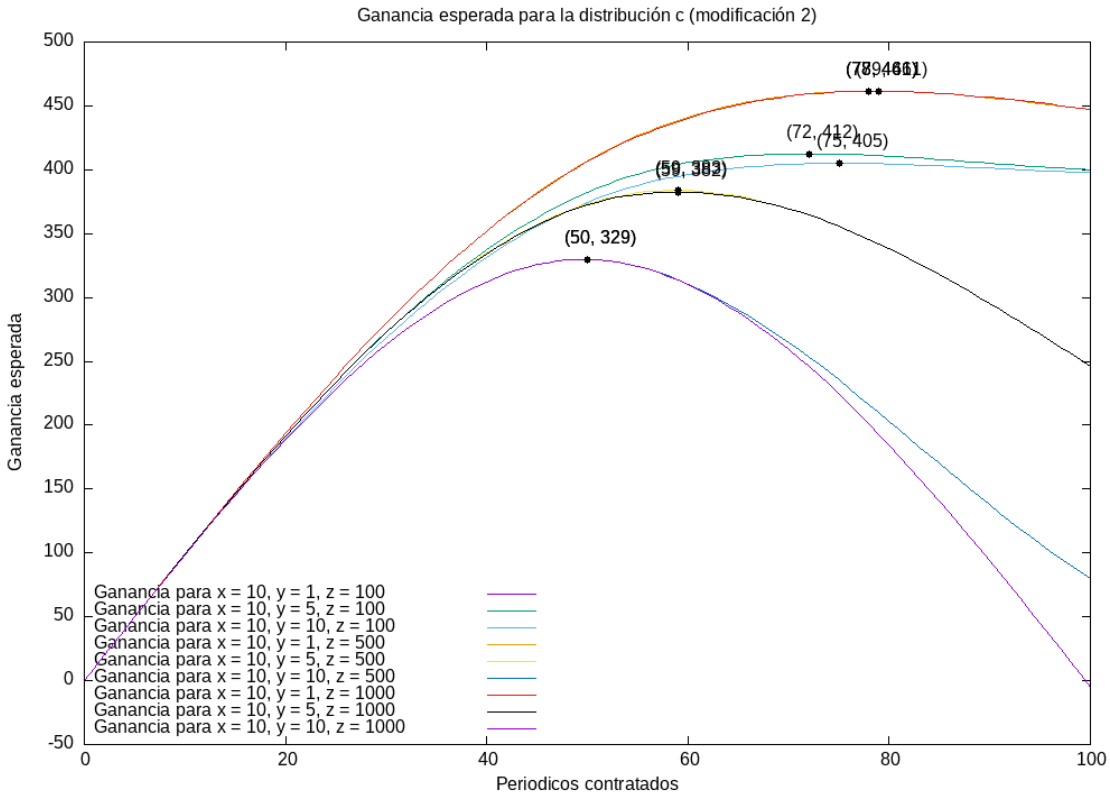
Podemos apreciar como en general, los resultados obtenidos se parecen significativamente a los obtenidos para la simulación sin modificar, por lo que podemos inferir, que resulta más económico fijar un precio de devolución variable segun el número de unidades que queden sin vender que fijar un precio total.

Esto tiene sentido, ya que al ser los precios de devolución por unidad más pequeños que los precios fijos de devolución, es decir, al ser y más pequeña que z, es muy improbable que la demanda sea lo suficientemente baja como para provocar unas pérdidas tan grandes por unidad no vendida que igualen o superen a las pérdidas producidas por un precio de devolución fijo.

Es también por este motivo que encontramos valores repetidos en las tablas y funciones superpuestas, ya que generalmente se aplica la función de ganancia utilizando el valor y en lugar del valor z.


Ganancia esperada para la distribución a (modificación 2)
Ganancia esperada
Periodicos contratados
(66, 325)
(49, 241)
Ganancia para x = 10, y = 1, z = 100
Ganancia para x = 10, y = 5, z = 100
Ganancia para x = 10, y = 10, z = 100
Ganancia para x = 10, y = 1, z = 500
Ganancia para x = 10, y = 5, z = 500
Ganancia para x = 10, y = 10, z = 500
Ganancia para x = 10, y = 1, z = 1000
Ganancia para x = 10, y = 5, z = 1000
Ganancia para x = 10, y = 10, z = 1000


Ganancia esperada para la distribución b (modificación 2)
Ganancia esperada
Periodicos contratados
(69, 280)
(42, 186)
(29, 131)
Ganancia para x = 10, y = 1, z = 100
Ganancia para x = 10, y = 5, z = 100
Ganancia para x = 10, y = 10, z = 100
Ganancia para x = 10, y = 1, z = 500
Ganancia para x = 10, y = 5, z = 500
Ganancia para x = 10, y = 10, z = 500
Ganancia para x = 10, y = 1, z = 1000
Ganancia para x = 10, y = 5, z = 1000
Ganancia para x = 10, y = 10, z = 1000


Ganancia esperada para la distribución c (modificación 2)
Ganancia esperada
Periodicos contratados
500
450
400
350
300
250
200
150
100
50
0
-50
0
20
40
60
80
100
(78, 461)
(72, 412)
(75, 405)
(50, 382)
(50, 329)
Ganancia para x = 10, y = 1, z = 100
Ganancia para x = 10, y = 5, z = 100
Ganancia para x = 10, y = 10, z = 100
Ganancia para x = 10, y = 1, z = 500
Ganancia para x = 10, y = 5, z = 500
Ganancia para x = 10, y = 10, z = 500
Ganancia para x = 10, y = 1, z = 1000
Ganancia para x = 10, y = 5, z = 1000
Ganancia para x = 10, y = 10, z = 1000

Apartado 2:
Generadores de datos

2.1: Mejorando los generadores

En este apartado se pedía introducir una serie de mejoras de eficiencia en los generadores de datos aleatorios.

Dichas mejoras son las siguientes

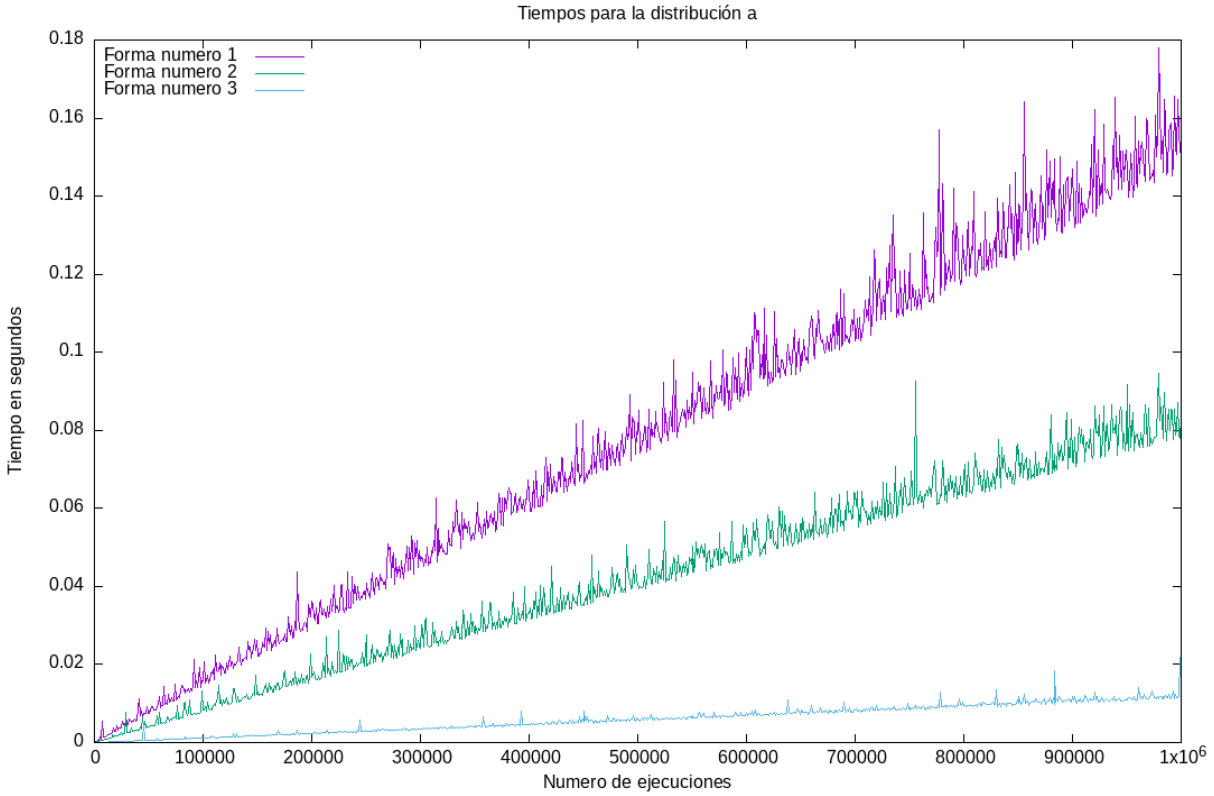
- Forma número 1: La forma número 1 es la de referencia, ya que es la manera en la que se han obtenido los datos en los apartados anteriores, es decir, recorriendo la tabla de probabilidad de forma secuencial.
- Forma número 2: Esta forma de mejora se basa en aplicar una búsqueda binaria para encontrar el valor aleatorio en la tabla de probabilidad.
- Forma número 3: Esta mejora es especifica para la distribución a y consigue obtener un tiempo constante de ejecución. Se basa en multiplicar por 100 el número obtenido entre 0 y 1 de manera uniforme. Dado que en la distribución a todos los valores poseen la misma probabilidad, simplemente multiplicando el valor uniforme aleatorio por el máximo, obtenemos el resultado requerido.
- Forma número 4: La forma número 4 de mejora es exclusiva para la distribución c y se basa en recorrer la tabla de probabilidad comenzando por los valores centrales y continuar en orden decreciente por la derecha y creciente por la izquierda. Así aseguramos comprobar primero los valores con mayor prioridad.

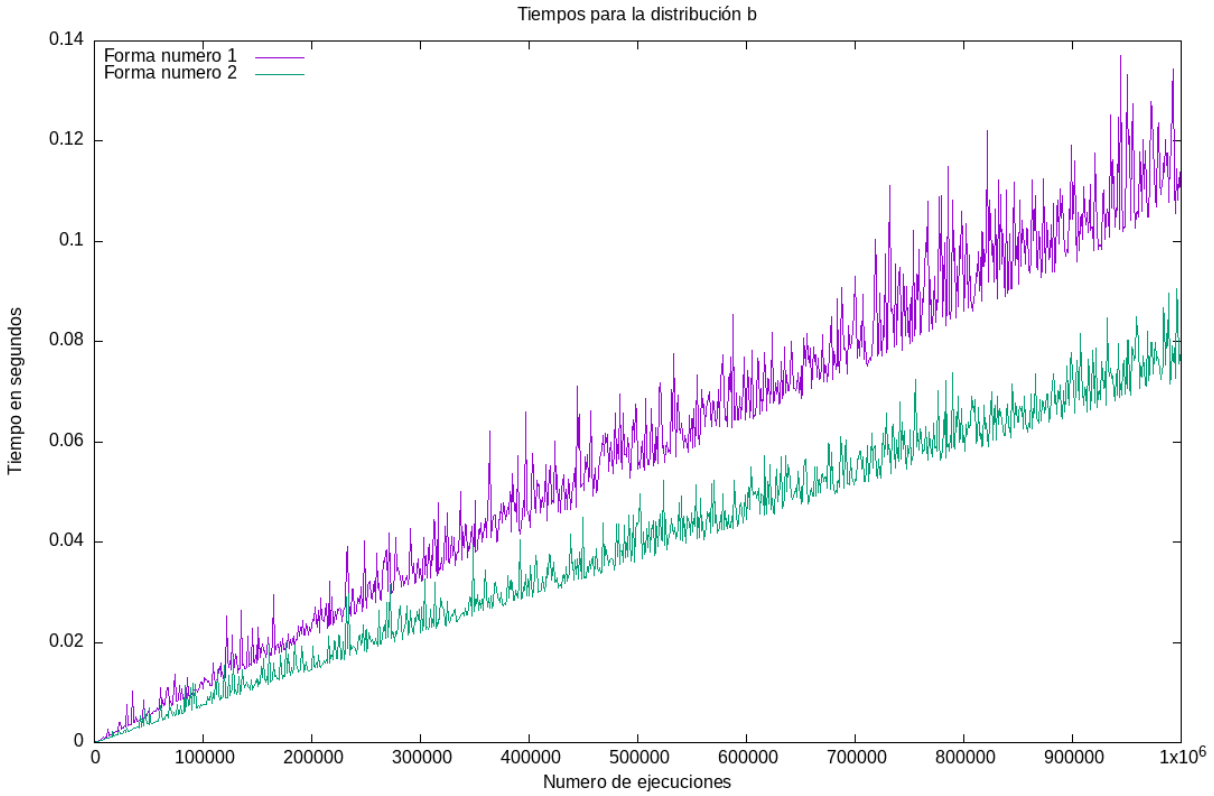
Podemos observar en las gráficas que se adjuntan a continuación:

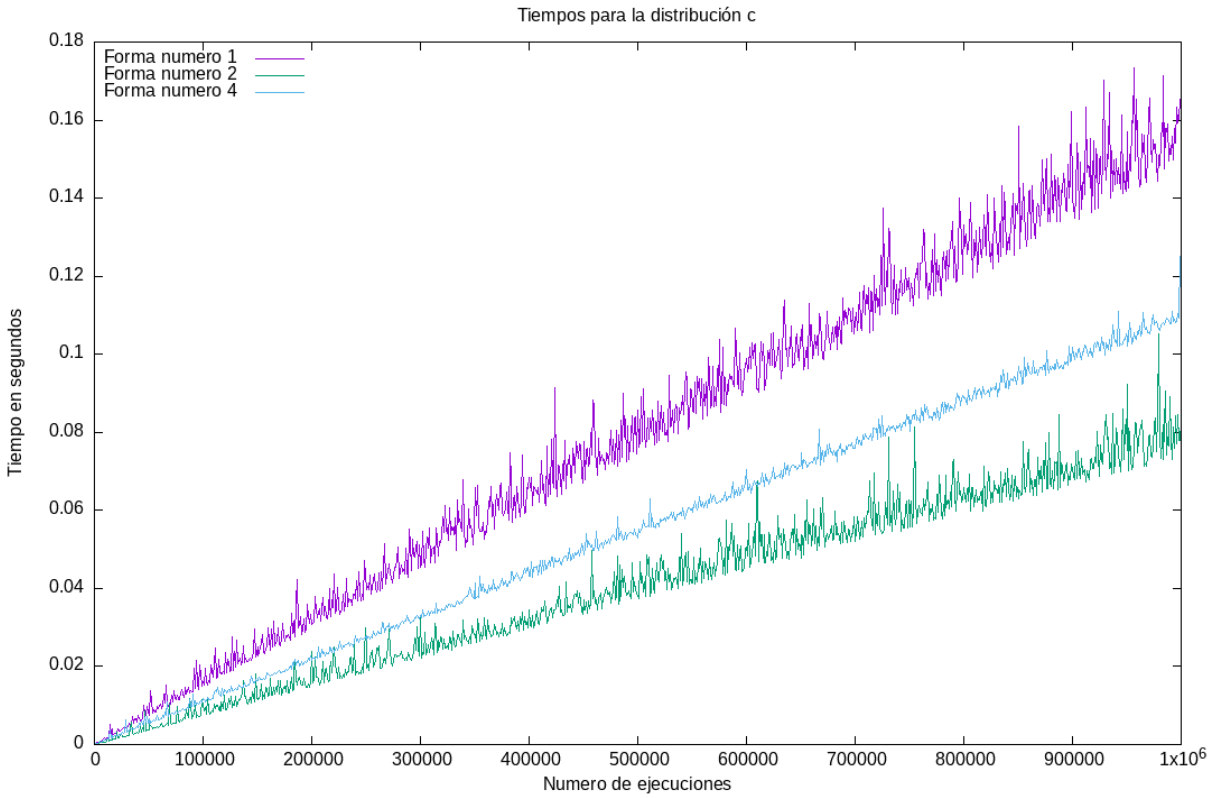
Para la distribución a, vemos como la mejora 2 — basada en búsqueda binaria — consigue mejorar en eficiencia a la forma 1, obteniendo resultados casi un 50 % más rápido. Sin embargo vemos como la mejora 4 es imbatible, cosa que tiene sentido, ya que es capaz de obtener resultados en tiempo constante sin importar el tamaño de la tabla de búsqueda.

Para la distribución b observamos que la mejora 2, como es lógico, obtiene mejores resultados a la hora de encontrar los valores deseados.

Para la distribución c, podemos ver que la mejora 4, es significativamente más eficiente que la forma normal — forma 1 —, pero es superada por la mejora 2.


Tiempos para la distribución a
Forma numero 1
Forma numero 2
Forma numero 3
Tiempo en segundos
Numero de ejecuciones
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0
100000
200000
300000
400000
500000
600000
700000
800000
900000
1x10^6


Tiempos para la distribución b
Forma numero 1
Forma numero 2
Tiempo en segundos
Numero de ejecuciones
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0
100000
200000
300000
400000
500000
600000
700000
800000
900000
1x10^6


Tiempos para la distribución c
Forma numero 1
Forma numero 2
Forma numero 4
Tiempo en segundos
Numero de ejecuciones
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0
100000
200000
300000
400000
500000
600000
700000
800000
900000
1x10^6

2.2: Generadores Congruenciales Lineales

2.2.1: Implementaciones

Para este aparado se pide que implementamos los siguientes generadores congruenciales lineales (g.c.l):

- $x_{n+1} = (2060x_n + 4321) \bmod m$
- $x_{n+1} = (2061x_n + 4321) \bmod m$

Para un $m = 10^4$

Para calcular los valores generados por estos g.c.l se han implementado los siguientes métodos:

- Aritmética entera:

```
return (a*x + c) % m;
```

- Aritmética real 'artesanal':

```
x = (a*x + c) / m;
x = (x - (int)x) * m;
return x;
```

- Aritmética real 'artesanal' corregida:

```
x = (a*x + c) / m;
x = (x - (int)x) * m;
x = (int)(x+0.5);
return x;
```

- Aritmética real usando fmod:

```
return fmod((a*x + c), m);
```

2.2.2: Resultados

En la siguiente tabla, podemos ver los periodos máximos obtenidos y para que semilla se ha obtenido con cada una de las implementaciones de los g.c.l:

Multiplicador	Método	Semilla	Periodo
2060	Aritmética entera	0	5
2060	Aritmética real 'artesanal' float	1253	193
2060	Aritmética real 'artesanal' double	268	2707
2060	Aritmética real 'artesanal' corregida float	5226	16
2060	Aritmética real 'artesanal' corregida double	0	5
2060	Aritmética real usando fmod	0	5
2061	Aritmética entera	0	10000
2061	Aritmética real 'artesanal' float	3196	283
2061	Aritmética real 'artesanal' double	0	10000
2061	Aritmética real 'artesanal' corregida float	6717	543
2061	Aritmética real 'artesanal' corregida double	0	10000
2061	Aritmética real usando fmod	0	10000

Primeramente debemos notar que el g.c.l que utiliza un multiplicador a = 2061 si ofrece el periodo máximo posible, es decir, tiene un periodo = m. Sin embargo el g.c.l con multiplicador a = 2060 no, la longitud máxima de su periodo es menor a m.

Esto es debido los siguientes factores:

- m y c son primos relativos, ya que el mcd(4321,10000) = 1.
- Para a = 2061, a - 1 es múltiplo de todos los primos que dividen a m, en este caso 1, 2 y 5. Sin embargo para a = 2060 esto no se cumple.
- Para a = 2061, a - 1 es múltiplo de 4, ya que m lo es también. Para a = 2060 esto no se cumple.

Por tanto esta comprobado que el g.c.l con multiplicador 2061 debe aportar un periodo máximo = m mientras que el g.c.l con multiplicador 2060 no.

También podemos notar que hay inconsistencias en la tabla, es decir, hay implementaciones que no aportan los resultados que deberían. A continuación analizaremos cada implementación por separado.

2.2.3: Aritmética entera

Multiplicador	Método	Semilla	Periodo
2060	Aritmética entera	0	5
2061	Aritmética entera	0	10000

En primer lugar analizaremos los resultados obtenidos para el método de aritmética entera.

Para un multiplicador de a = 2060 vemos que el periodo máximo que ofrece el g.c.l es 5, mientras que para un multiplicador de a = 2061 alcanza el máximo.

Los resultados que aporta esta implementación no son para nada sorprendentes, pues simplemente aplica la formula clásica para calcular g.c.l y además trabaja con valores de tipo entero, lo que elimina cualquier problema de precisión.

2.2.4: Aritmética real 'artesanal'

Multiplicador	Método	Semilla	Periodo
2060	Aritmética real 'artesanal' float	1253	193
2060	Aritmética real 'artesanal' double	268	2707
2061	Aritmética real 'artesanal' float	3196	283
2061	Aritmética real 'artesanal' double	0	10000

Como se puede ver, en esta implementación los resultados son algo más variopintos.

Llamamos artesanal a esta implementación puesto que nosotros tratamos con los decimales manualmente, en lugar de aplicar alguna función de redondeo o truncamiento. Se han programado dos versiones de esta implementación una que trabaja con valores float y otra que trabaja con valores double.

El problema principal en esta implementación es que no se devuelve la parte entera del número, puesto que en principio no debería hacer falta, ya que las divisiones deberían ser enteras y exactas. Sin embargo, no lo son.

En cada división se producen una serie de errores de precisión derivados de la aritmética en coma flotante, que alteran los números de manera que al principio el error es pequeño, pero división tras división se va agravando más y más hasta alterar por completo los números generados por el g.c.l

Además podemos ver que según utilicemos valores de tipo float o double, los errores de precisión no son los mismos, puesto que el tipo double posee el doble de precisión que el tipo float.

Finalmente este generador es capaz de producir números con decimales, cuando un g.c.l debe producir números enteros. Por esta razón parece que el g.c.l con a = 2061 y tipo double parece que genera 10000 números distintos. Lo que sucede es que se ha limitado al generador a producir solo 10000 valores, puesto que si no al generar números decimales, el generador podría haber ciclado durante una cantidad de tiempo considerable.

2.2.5: Aritmética real 'artesanal' corregida

Multiplicador	Método	Semilla	Periodo
2060	Aritmética real 'artesanal' corregida float	5226	16
2060	Aritmética real 'artesanal' corregida double	0	5
2061	Aritmética real 'artesanal' corregida float	6717	543
2061	Aritmética real 'artesanal' corregida double	0	10000

Este método es parecido al anterior, ya que también es un método artesanal. Sin embargo, en esta implementación se consiguen subsanar los errores de precisión de manera bastante buena.

Si bien es cierto que se siguen produciendo si trabajamos con valores de tipo float, las divisiones con valores double son lo suficientemente precisas para que el redondeo funcione.

Este método realiza los mismos cálculos que el anterior salvo porque al final suma 0.5 al valor obtenido — para conseguir arreglar el error de precisión — y el cambio fundamental es que se queda con la parte entera de la suma realizada, por lo que elimina la parte decimal de todo número generado.

2.2.6: Aritmética entera usando fmod

Multiplicador	Método	Semilla	Periodo
2060	Aritmética real usando fmod	0	5
2061	Aritmética real usando fmod	0	10000

Esta implementación sencillamente hace uso de la función fmod, que permite realizar la operación módulo para valores de tipo real como float y double. Al utilizar una función de la biblioteca cmath, nos olvidamos de tener que prestar atención a los errores de precisión

2.2.7: Conclusión

Finalmente y como conclusión creo que la mejor forma de implementar un g.c.l es hacer una correcta gestión de los errores de precisión o trabajar con números enteros y evitar estos errores directamente.

Los generadores de datos son esenciales a la hora de simular un sistema de manera realista y para poder realizar buenas estimaciones, por tanto, han de ser diseñados de manera que no produzcan fallos del tipo que hemos comentado.