CIMAT

Reporte de proyecto final de Optimización

01 de junio de 2023

Autores: *Alondra Elizabeth Matos Mendoza, María Guadalupe López Salomón*

Maestría en Cómputo Estadístico, CIMAT Sede Monterrey

Resolviendo un problema de optimización de portafolios con restricción de cardinalidad a través de un algoritmo genético

Introducción

En un mercado determinado, los inversores se encuentran frente a diferentes activos o valores que ofrecen diversos niveles de rendimiento según el riesgo subyacente. Un aspecto práctico al que se enfrentan los inversores racionales es cómo seleccionar el activo o la combinación de activos para construir un portafolio óptimo, minimizando el riego en algún nivel de retorno.

Harry Markowitz (1952) propuso que el rendimiento de la cartera debe medirse en dos dimensiones distintas: la media que describe el rendimiento esperado y el riesgo que mide la incertidumbre del rendimiento. Para un universo particular de activos, el conjunto de carteras de activos que ofrecen el menor riesgo de una inversión sujeto al retorno esperado del inversionista constituyen la frontera eficiente.

Markowitz utilizó la varianza de la rentabilidad como medida del riesgo. De esta manera, el problema de selección de cartera se reduce a encontrar los portafolios eficentes mediante programación cuadrática. Las soluciones obtenidas son óptimas y el proceso de selección de estas soluciones puede estar limitado por consideraciones prácticas que pueden expresarse como restricciones lineales.

Sin embargo, no es fácil añadir restricciones enteras que limitan una cartera para tener un número específico de activos, o para imponer límites a la proporción de la cartera en un activo dado (si se mantiene alguno de los activos).

El presente documento tiene como finalidad examinar el algoritmo genético propuesto por *Chang et al.*[1] para la selección de portafolios con la imposición de restricciones enteras.

Descripción del problema

Una cartera de inversiones se define por el vector $w = (w_1, \ldots, w_n)$, donde w_j denota la proporción de la inversión a invertir en el activo j. Sea μ_j el retorno esperado del activo j, el rendimiento esperado de la cartera es $\mu^T w$.

Sea Σ la matriz de varianzas y covarianzas del retorno de los activos. La entrada $\Sigma_{j,j}$ es la varianza del rendimiento del activo j. Una alta varianza indica una volatilidad o riesgo alto, mientras que una varianza baja refleja estabilidad o bajo riesgo. La entrada $\Sigma_{i,j}$ es la covarianza de los rendimientos de los activos i y j. Un valor positivo de $\Sigma_{i,j}$ indica activos cuyos valores suelen moverse en la misma dirección, como suele ocurrir con las acciones de empresas del mismo sector.Un valor negativo indica activos cuyos valores generalmente se mueven en direcciones opuestas. El riesgo del portafolio es la varianza esperada $w^T \Sigma w$.

El problema de optimización tiene dos objetivos: maximizar el retorno $\mu^T w$. y minimizar el riesgo $w^T \Sigma w$. La importancia de estos objetivos diferirá en función de la tolerancia al riesgo del inversor. Introduciendo un parámetro λ $(0 \leq \lambda \leq 1)$, que representa la voluntad del inversor de equilibrar el riesgo y la rentabilidad, la función objetivo es una combinación de estos dos objetivos.

Considerando los parámetros:

- N: número de activos disponibles.
- K : número deseado de activos en el portafolio.
- ϵ_i: proporción mínima que se debe mantener del activo i si el activo i se mantiene. Representa el nivel mínimo de transacción.
- δ_i: proporción máxima que se debe mantener del activo i si el activo i se mantiene. Limita la exposición de la cartera al activo i.

- $z_i = \begin{cases} 1 & \text{si se mantiene el activo i} \\ 0 & \text{en otro caso} \end{cases}$

Y sean las variables de decisición w_i la ponderación que recibe cada activo i $(i = 1, ..., N)$ en el portafolio, el modelo de optimización de cartera con restricciones de cardinalidad es:

Minimizar

$$\lambda \left[w^T \Sigma w\right] - (1 - \lambda) \left[\mu^T w\right] \tag{1}$$

Sujeto a:

$$\sum_{i=1}^{N} w_i = 1 \tag{2}$$

$$\sum_{i=1}^{N} z_i = K \tag{3}$$

$$\epsilon_i z_i \leq w_i \leq \delta_i z_i, \quad i = 1, ..., N \tag{4}$$

$$z_i \in [0, 1], \quad i = 1, ..., N \tag{5}$$

en donde $0 \leq \epsilon_i \leq \delta_i \leq 1$, $w^T \Sigma w = \sum_i^N \sum_j^N w_i w_j \sigma_{ij}, \mu^T w = \sum_i^N w_i \mu_i$

La Ec. 3 asegura que se mantengan exactamente K activos. La Ec. 4 asegura que si el activo i se mantiene $(z_i = 1)$, su proporción w_i debe estar entre ϵ_i y δ_i, mientras que si no se mantiene el activo i $(z_i = 0)$, su proporción w_i es igual a 0.

En el caso $\lambda = 0$, la Ec. 1 representa maximizar el rendimiento esperado independientemente del riesgo involucrado En el caso $\lambda = 1$, la Ec. 1 representa minimizar el riesgo independientemente del rendimiento involucrado.

En particular, si se establece $K = N, \epsilon_i = 0$ y $\delta_i = 1 (i = 1, \dots N)$, se calcula la frontera eficiente sin restricciones.

Metodología

El algoritmo genético (GA, por sus siglas en ingés) es un método heurístico introducido por John Holland en 1960 y se basa en conceptos genéticos y mecanismos de selección natural.

Dicha heurística es un algoritmo iterativo diseñado para encontrar una solución óptima. Opera sobre una población de tamaño fijo, que consta de puntos candidatos conocidos como cromosomas. El tamaño de población fijo crea competencia entre los cromosomas. Cada cromosoma representa una solución potencial codificada como un conjunto de elementos llamados genes. En cada iteración, denominada generación, se crea una nueva población con el mismo número de cromosomas. Esta nueva población consiste en cromosomas que están mejor "adaptados" a su entorno, según lo determinado por una función selectiva. Con cada generación, los cromosomas tienden a acercarse al óptimo de la función selectiva. La creación de una nueva población a partir de la anterior implica la aplicación de operadores genéticos, a saber, selección, cruce y mutación. Estos operadores son de naturaleza estocástica. El proceso de selección es el paso inicial en un algoritmo genético, donde el algoritmo elige los factores más relevantes que optimizan la función objetivo. El cruce permite que dos cromosomas "padres"seleccionados generen nuevos cromosomas "hijos". La mutación implica la alteración de uno o más genes en un cromosoma por inversión u otros medios.[2]

La Figura 1 presenta los pasos fundamentales de un algoritmo genético simple.

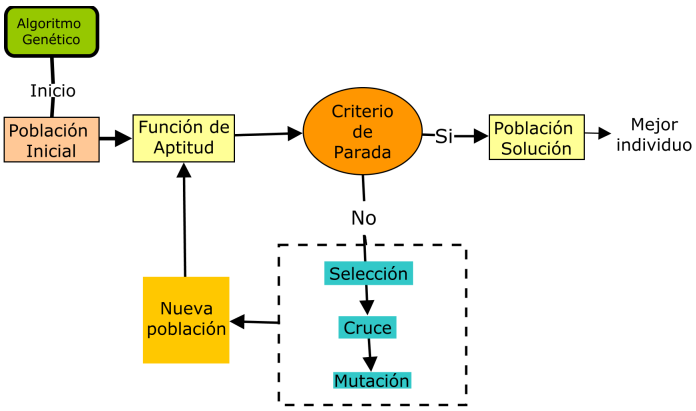

Figura 1: Diagrama de flujo del algoritmo genético

Se ha demostrado que la optimización de carteras utilizando algoritmos genéticos proporciona resultados superiores en comparación con otros métodos heurísticos[3].

En el algoritmo genético descrito en [1], la representación cromosómica de una solución tiene dos partes: un conjunto Q de K distintos activos y K números reales s_i $(0 \leq s_i \leq 1)$ $i \in Q$. Dado un conjunto Q de K activos, una fracción $\sum_{j \in Q} \epsilon_j$ de la cartera total ya está contabilizada, por lo que s_i es la parte de la proporción libre de la cartera $(1 - \sum_{i \in Q} \epsilon_i)$ asociada al activo $i \in Q$.

No todos los posibles cromosomas son soluciones factibles debido a la restricción de los límites de la proporción de un activo. Sin embargo, en el proceso de evaluar la aptitud de cada cromosoma, se garantiza que la solución evaluada sea factible mediante el uso del algoritmo de evaluación, cuyo pseudocódigo se encuentra detallado en el Anexo 1. Dicho algoritmo es una función que devuelve, para cada posible solución, el valor de la aptitud y el valor reestablecido de s_i de los activos que conforman la solución.

A continuación, se describe la esencia del algoritmo de evaluación:

1. Si $\sum_{i \in Q} \epsilon_i < 1$ o $\sum_{i \in Q} \delta_i < 1$, el conjunto de posibles soluciones Q no es factible. Para dicho conjunto, se establece el valor de la función objetivo igual a ∞

2. Se calculan las proporciones w_i, de tal forma que

$$w_i = \epsilon_i + s_i F/L$$

donde $L = \sum_{i \in Q} s_i$ y F es la proporción libre del portafolio, $F = 1 - \sum_{i \in Q} \epsilon_i$.

3. Mientras haya algún activo i en Q tal que $w_i > \delta_i$, se recalcula w_i para toda i siguiendo el procedimiento que se muestra a continuación:

   a) Obtener la suma actual de s_i:

$$L = \sum_{i:w_i \leq \delta_i} s_i$$

   para todos los activos i tal que $w_i \leq \delta_i$

   b) Obtener la proporción libre del portafolio:

$$F = 1 - \sum_{i:w_i \leq \delta_i} \epsilon_i + \sum_{i:w_i > \delta_i} \delta_i$$

   c) Establecer la proporción del activo i tal que $w_i \leq \delta_i$ como:

$$w_i = \epsilon_i + s_i F/L$$

   d) Establecer la proporción del activo i tal que $w_i > \delta_i$ como:

$$w_i = \delta_i$$

4. Se calcula la aptitud (valor de la función objetivo):

$$f = \lambda \left[\sum_{i \in Q} \sum_{j \in Q} w_i w_j \sigma_{ij} \right] - (1 - \lambda) \left[\sum_{i \in Q} w_i \mu_i \right]$$

5. Se reestablece s_i para cada activo del conjunto Q:

$$s_i = w_i - \epsilon_i, \ \forall i \in Q$$

Los pasos que describen el algoritmo genético desarrollado por *Chang et al.*[1] son:

Para un valor de λ dado:

1. Generar una población inicial de cromosomas de tamaño p .
2. Evaluar la aptitud de los cromosomas de la población. La aptitud es el valor de la función objetivo.
3. Realizar el siguiente proceso de forma repetida hasta alcanzar el límite máximo de iteraciones:

   a) Elegir a los padres de la población por el método de selección de torneo binario.

   Este proceso involucra la creación de dos grupos, cada uno con dos individuos elegidos al azar de la población. Los individuos con la mayor aptitud (menor valor de la función objetivo) de cada grupo serán elegidos para ser los padres.

   b) Aplicar el cruce de los padres mediante el método uniforme para crear un único hijo. Cada activo se elige de cualquiera de los padres con igual probabilidad. Si el activo i está presente en ambos padres, también estará presente en el hijo, con un valor s_i seleccionado al azar de cualquiera de los padres. Si el activo i está presente en solo uno de los padres, hay una probabilidad de 0.5 de que esté presente en el hijo.

   c) Mutar el cromosoma del hijo creado.

   En el proceso de mutación, el valor s_i de un activo i (elegido aleatoriamente) incrementa o decrementa un 10 % el valor $\epsilon_i + s_i$ de manera equiprobable, es decir:

   Con probabilidad 0.5,

$$s_i = 0.9(\epsilon_i + s_i) - \epsilon_i$$

   o bien

$$s_i = 1.1(\epsilon_i + s_i) - \epsilon_i$$

   En caso de que s_i sea negativo, eliminar el activo i del conjunto de K activos diferentes correspondiente al cromosoma del hijo.

   d) Mientras la cardinalidad del conjunto de activos correspondiente al cromosoma del hijo sea mayor a K, eliminar del conjunto el activo con el valor más pequeño de s_i

   e) Mientras la cardinalidad del conjunto de activos correspondiente al cromosoma del hijo sea menor a K, añadir si es posible un activo j tomado aleatoriamente del conjunto de activos no heredados de los padres. De lo contrario, elegir aleatoriamente un activo j (que no tenga ya el hijo) del conjunto total de activos posibles y establecer $s_j = 0$

   f) Evaluar la aptitud del hijo

   g) Reemplazar al peor individuo de la población por el hijo. El peor individuo es aquél con la menor aptitud, es decir, el valor más grande obtenido en la función objetivo.

4. Obtener al mejor individuo de la población mejorada.

El pseudocódigo de este algoritmo genético se muestra en el Anexo 2.

Experimentación computacional

En la presente sección, se describe la experimentación computacional realizada para llevar a cabo la implementación de la metodología propuesta. A continuación, se detallan los componentes del ambiente computacional utilizado, incluyendo el sistema operativo, la memoria RAM del equipo y el software utilizado para implementar el código.

La experimentación computacional se llevó a cabo en un entorno basado en macOS Monterrey Versión 12.3, con un equipo cuyo procesador corresponde al Apple M1 Pro y 16 GB de memoria RAM. Se utilizó Google Colab como el entorno de desarrollo en línea, con Python 3 como el lenguaje de programación principal. Estas configuraciones proporcionaron un ambiente adecuado y eficiente para llevar a cabo la implementación de la metodología propuesta.

Para probar la heurística, se consideró la base de datos de acciones correspondientes al índice de mercado 'Hang Seng (Hong Kong)', con $N = 31$, del cuál se extrajo los precios semanales en un periodo que comprende de Marzo de 1992 a Septiembre de 1997, y a partir de los cuales se calcularon los rendimientos y covarianzas semanales. Cabe señalar que las acciones con valores faltantes se abandonaron. Es importante destacar que esta base de datos considerada fue usada e implementada en [1], como puede verificarse en la biblioteca *O-R Library, J. E Beasley.*

Analogamente, se consideró una segunda base de datos, para la cual se consideraron los siguientes símbolos: ['AAPL', 'GOOGL', 'MSFT','DAX', 'AMZN', 'JPM', 'V', 'JNJ', 'WMT', 'PG','MA', 'UNH', 'NVDA', 'VZ', 'DIS', 'HD', 'BAC', 'T', 'PYPL', 'PFE','INTC', 'ADBE', 'TSLA', 'CMCSA', 'KO', 'MRK', 'NFLX', 'CRM', 'CSCO', 'XOM','PEP', 'ABBV', 'ABT', 'CVX', 'ORCL', 'MCD', 'TSM', 'IBM', 'HON','AMGN','NKE', 'ACN', 'TXN', 'SAP', 'BMY','SIEGY', 'TM', 'BA', 'GSK', 'SNY'], correspondientes a acciones que cotizan en diferentes bolsas de valores. La lista detallada de las empresas a las que corresponden los símbolos se muestran en el Anexo 3. Esta segunda base de datos se extrajo de la librería de `yfinance` considerando los periodos que comprenden del primero de Enero de 2019 al primero de Enero de 2022, con la consideración de suprimir las acciones con valores faltantes. Se obtuvieron los rendimientos esperados y covarianzas diarias para $N = 50$.

Para la primera instancia se probó el rendimiento del algoritmo 5 veces. Con $N = 31, E = 5, K = 10$, $\epsilon_i = 0, \delta_i = 0$ $(i = 1, ..., N)$ y un criterio de parada de iteraciones máximas correspondientes a $max_{iter} = 31000$. Los datos obtenidos se exhiben en la figura (2) del Anexo 4, donde se muestran los valores promedio de la función objetivo para distintos valores de λ, así como los diferentes tiempos de ejecución en cada evaluación.

De manera similar, para la segunda instancia, se probó el rendimiento del algoritmo 5 veces. Considerando valores de $N = 50, E = 5, K = 10$, $\epsilon_i = 0, \delta_i = 0$ $(i = 1, ..., N)$ y un criterio de parada de iteraciones máximas correspondientes a $max_{iter} = 50000$. Los resultados obtenidos se exhiben en la figura (3) del Anexo 4. Se obtuvieron los valores promedio de la función objetivo para cada valor de λ, así como los promedios del tiempo de ejecución en cada evaluación.

Conclusiones

A partir de los resultados registrados en el Anexo 4, se observó que para la primera instancia, es decir, para $N = 31$, los valores de la función objetivo se mantuvieron estables a lo largo de las evaluaciones, no mostrando cambios significativos. Los valores de la función objetivo variaron en un rango muy estrecho en cada evaluación. De manera similar, para la segunda instancia $N = 50$, se observó una consistencia similar, con valores de la función objetivo fluctuando ligeramente en cada evaluación. Además, los tiempos de ejecución del algoritmo también se mantuvieron consistentes en ambas instancias, con un promedio de 96.8 y 156.8 segundos respectivamente por cada evaluación.

Estos resultados indican que el algoritmo implementado ha convergido a una solución estable y consistente. Después de varias iteraciones, los individuos en la población han alcanzado un equilibrio y la función objetivo se ha estabilizado en valores similares. La consistencia en los resultados se refuerza al repetir el proceso de evaluación y obtener valores de función objetivo y tiempos de ejecución similares en cada instancia.

Estos hallazgos son prometedores y sugieren que el algoritmo genera resultados reproducibles y confiables. La estabilidad y consistencia en la función objetivo indican que el algoritmo está encontrando una solución óptima o muy cercana a ella. Sin embargo, es importante destacar que este estudio se centra en bases de datos específicas y pueden haber limitaciones en la generalización de los resultados.

Si bien los tiempos de ejecución fueron consistentes en las evaluaciones realizadas, se notó que podrían optimizarse aún más mediante una mejora en la implementación del algoritmo, a través de una estrategia de programación más eficiente, como el uso de bucles y estructuras de datos simplificadas o mediante técnicas de programación paralela.

Aunado a la captura de los datos proporcionados por el algoritmo, se procedió a graficar la frontera eficiente de la cartera de activos para la estancia correspondiente a $N = 31$, considerando $E = 50$. Es importante destacar que al establecer $N = K, \epsilon = 0$ y $\delta = 1$, la frontera eficiente obtenida corresponde a la frontera eficiente para el caso sin restricciones. Esta frontera se puede obtener mediante programación cuadrática; por lo que, para comparar el desempeño del algoritmo génetico, se resolvió el problema de optimización cuadrática para obtener la frontera eficiente real. Am-

bos resultados se pueden apreciar en el Anexo 5.

Al graficar ambas fronteras eficientes, se compararon visualmente para evaluar su similitud. Se observó que la frontera eficiente generada por el algoritmo y la obtenida mediante programación cuadrática eran muy similares en forma y tendencia. Esto respalda la validez y confiabilidad de la frontera eficiente obtenida por el algoritmo implementado.

Estos resultados corroboran que el algoritmo ha sido efectivo en la generación de la frontera eficiente de la cartera de activos. Además, el uso de la programación cuadrática como método de validación permitió confirmar que la frontera obtenida por el algoritmo coincide con la solución óptima.

Con base en estos hallazgos, se puede concluir que el algoritmo implementado es una herramienta confiable y eficiente para la generación de la frontera eficiente de la cartera de activos. Este resultado es de gran importancia en la toma de decisiones financieras, ya que proporciona información valiosa sobre las combinaciones óptimas de activos para maximizar el rendimiento y minimizar el riesgo.

Referencias

[1] Chang, T. J., Meade, N., Beasley, J. E., & Sharaiha, Y. M.(2000). *"Heuristics for cardinality constrained portfolio optimisation"*. Computers & Operations Research, 27(13), 1271-1302.

[2] Guennoun, Z., & Hamza, F(2012). *"Stocks portfolio optimization using classification and genetic algorithms"*.Applied Mathematical Sciences, 6(94), 4673-4684.

[3] Ackora-Prah, J., Gyamerah, S. A., & Andam, P. S.(2014). *.A heuristic crossover for portfolio selection".*

Anexo 1.

Algorithm 1 Evaluación

evaluate (S, λ)
S es la solución actual y consiste de:
Q es el conjunto de K activos distintos en la solución actual
s_i es el valor actual para el activo $i \in Q$
λ es el parámetro de ponderación actual
f es el valor regresado de la función objetivo para la solución actual S
w_i es la proporción asociada con cada activo $i \in Q$

begin
$f := \infty$
if $\sum_{i \in Q} \epsilon_i > 1$ o $\sum_{i \in Q} \delta_i < 1$ **then** ▷ No factible
return
end if
$L := \sum_{i \in Q} S_i$ ▷ L es la suma actual s_i
$F := 1 - \sum_{i \in Q} \epsilon_i$ ▷ F es la proporción libre
$w_i := \epsilon_i + s_i F / L \ \forall i \in Q$ ▷ Calcula las proporciones para satisfacer ϵ_i y sumar hasta 1
▷ Procedimiento iterativo que satisface las proporciones máximas

$R := \emptyset$ ▷ R es un conjunto de i cuyas proporciones se ajustan en δ_i

while exista $i \in Q - R$ con $w_i > \delta_i$ **do** ▷ Iterar hasta que sea factible
for all $i \in Q - R$ **do**
if $w_i > \delta_i$ **then**
$R := R \cup [i]$ ▷ si w_i es mayor que δ_i añadir a R
end if
$L := \sum_{i \in Q - R} s_i$ ▷ L es la suma actual S_i
$F := 1 - \left(\sum_{i \in Q - R} \epsilon_i + \sum_{i \in R} \delta_i \right)$ ▷ F es la proporción libre
$w_i := \delta_i \ \forall i \in R$ ▷ Proporciones para $i \in R$

end for
end while
$f := \lambda \left[\sum_{i \in Q} \sum_{j \in Q} w_i w_j \sigma_{ij} \right] - (1 - \lambda) \left[\sum_{i \in Q} w_i \mu_i \right]$ ▷ Solución factible
$s_i := w_i - \epsilon_i \ \forall i \in Q$ ▷ Reestablecer s_i

end

Anexo 2

Algorithm 2 Algoritmo genético

Require: E es el número de λ valores que deseamos examinar
P es la población
S^*, S^{**} son dos soluciones (padres) seleccionadas de la población para emparejar
C es el hijo de S^* and S^{**} y consiste de R el conjunto de K distintos activos en C
c_i el valor del activo $i \in R$.
A^* es un conjunto de activos que están en los padres, pero no en los hijos (junto con los valores asociados)
T^* número de iteraciones
begin

for $e := 1$ *to* **do**

$\lambda := (e-1)/(E-1)$ ▷ examinar E λ valores igualmente espaciados en $[0,1]$
initialise $P := \{S_1, ..., S_{100}\}$ ▷ inicialización aleatoria, exactamente K activos en cada S_p
evaluacion $(S_p, \lambda)\, p = 1, ..., 100$ ▷ evaluar las soluciones
for $t := 1$ to T^* **do** ▷ T^* iteraciones en todo
seleccionar S^*, $S^{**} \in P$ por el método de torneo binario
cruza $C := (S^*, S^{**})$ por el método uniforme
$A^* := S^* \cup S^{**} - C$ ▷ encontrar activos en los padres, no en el hijo
elegir aleatoriamente $i \in R$ y $m = 1$ o 2
if $m = 1$ **then**
$c_i := 0.9(\epsilon_i + S_i)$
else
$c_i := 1.1(\epsilon_i + S_i) - \epsilon_i$ ▷ mutación
end if
if $c_i < 0$ **then**
$R := R - [i]$ ▷ si es necesario, borrar activo i

end if
▷ asegurar que C solo tenga K activos
while $|R| > K$ **do**
borrar el activo $j \in R$ con el c_j más pequeño de R
end while
while $|R| < K$ **do**
if $|A^*| \neq 0$ **then** ▷ agregar activo de un padre si es posible
agregar a C un activo j de A^* elegido aleatoriamente ▷ agregar activo
$A^* := A^* - [j]$
else
agregar a C un activo $j \notin R$ elegido aleatoreamente y establecer $c_j := 0$
end if
end while
▷ C tiene ahora exactamente K activos
evaluacion (C, λ)
encontrar j tal que $f(S_j) = max[f(S_p)|p = 1, ..., 100]$ ▷ encontrar el peor miembro de la población
$S_j := C$ ▷ reemplazar el peor miembro con un hijo
end for
end for
end

Anexo 3. Empresas correspondientes a los símbolos de acciones cotizadas en diferentes bolsas de valores implementados como base de datos en la metodología del presente informe:

- AAPL: Apple Inc.
- GOOGL: Alphabet Inc. (empresa matriz de Google)
- MSFT: Microsoft Corporation
- DAX: DAX 30 (índice bursátil alemán)
- AMZN: Amazon.com, Inc.
- JPM: JPMorgan Chase Co.
- V: Visa Inc.
- JNJ: Johnson Johnson
- WMT: Walmart Inc.
- PG: Procter Gamble Company
- MA: Mastercard Incorporated
- UNH: UnitedHealth Group Incorporated
- NVDA: NVIDIA Corporation
- VZ: Verizon Communications Inc.
- DIS: The Walt Disney Company
- HD: The Home Depot, Inc.
- BAC: Bank of America Corporation
- T: ATT Inc.
- PYPL: PayPal Holdings, Inc.
- PFE: Pfizer Inc.
- INTC: Intel Corporation
- ADBE: Adobe Inc.
- TSLA: Tesla, Inc.
- CMCSA: Comcast Corporation
- KO: The Coca-Cola Company
- MRK: Merck Co., Inc.
- NFLX: Netflix, Inc.
- CRM: Salesforce.com, Inc.
- CSCO: Cisco Systems, Inc.
- XOM: Exxon Mobil Corporation
- PEP: PepsiCo, Inc.
- ABBV: AbbVie Inc.
- ABT: Abbott Laboratories
- CVX: Chevron Corporation
- ORCL: Oracle Corporation
- MCD: McDonald's Corporation
- TSM: Taiwan Semiconductor Manufacturing Company Limited
- IBM: International Business Machines Corporation
- HON: Honeywell International Inc.

- AMGN: Amgen Inc.
- NKE: Nike, Inc.
- ACN: Accenture plc
- TXN: Texas Instruments Incorporated
- SAP: SAP SE
- BMY: Bristol-Myers Squibb Company
- SIEGY: Siemens AG
- TM: Toyota Motor Corporation
- BA: The Boeing Company
- GSK: GlaxoSmithKline plc
- SNY: Sanofi

Anexo 4.

	Valor de la función objetivo					**Time(s)**		
Evaluaciones	$\lambda_1 = 0.0$	$\lambda_2 = 0.25$	$\lambda_3 = 0.5$	$\lambda_4 = 0.75$	$\lambda_5 = 1$	**CPU**	**Wall**	**Total**
Primera	-0.010865	0.08825213	0.18125031	0.27482704	0.36717278	95	98	99
Segunda	-0.010865	0.08842956	0.18166007	0.27473444	0.367284	94	97	98
Tercera	-0.010865	0.08840496	0.1813265	0.2743189	0.3676613	93	96	97
Cuarta	-0.010865	0.08839864	0.18129243	0.27466946	0.36781126	93	96	97
Quinta	-0.010865	0.08842289	0.18125573	0.27421884	0.3677441	94	97	98
PROMEDIO	**-0.010865**	**0.088381636**	**0.181357008**	**0.274553736**	**0.367534688**	**93.8**	**96.8**	**97.8**

Figura 2: Desempeño del algoritmo para $N = 31$

	Valor de la función objetivo					**Time(s)**		
Evaluaciones	$\lambda_1 = 0.0$	$\lambda_2 = 0.25$	$\lambda_3 = 0.5$	$\lambda_4 = 0.75$	$\lambda_5 = 1$	CPU	Wall	Total
Primera	-0.00464538	-0.00303877	-0.00152619	-0.00036465	9.30E-05	154	158	160
Segunda	-0.00464538	-0.00303877	-0.00152619	-0.00037551	9.31E-05	153	157	160
Tercera	-0.00464538	-0.00303877	-0.00152619	-0.00037862	9.32E-05	153	157	160
Cuarta	-0.00464538	-0.00303877	-0.00152619	-0.00037869	9.37E-05	152	157	159
Quinta	-0.00464538	-0.00303877	-0.00152619	-0.00037869	9.31E-05	151	155	158
PROMEDIO	**-0.00464538**	**-0.00303877**	**-0.00152619**	**-0.000375232**	**9.32E-05**	**152.6**	**156.8**	**159.4**

Figura 3: Desempeño del algoritmo para $N = 50$

Anexo 5. Se muestran las gráficas de la frontera eficiente obtenida a través del algoritmo genético para $N = 31$ y la frontera eficiente real obtenida al resolver el problema de programación cuadrática.

El color azul en la gráfica corresponde a la frontera eficiente real sin restricciones obtenida mediante programación cuadrática y los puntos rojos corresponden a la frontera eficiente para el caso sin restricciones obtenida mediante el algoritmo genético implementado.

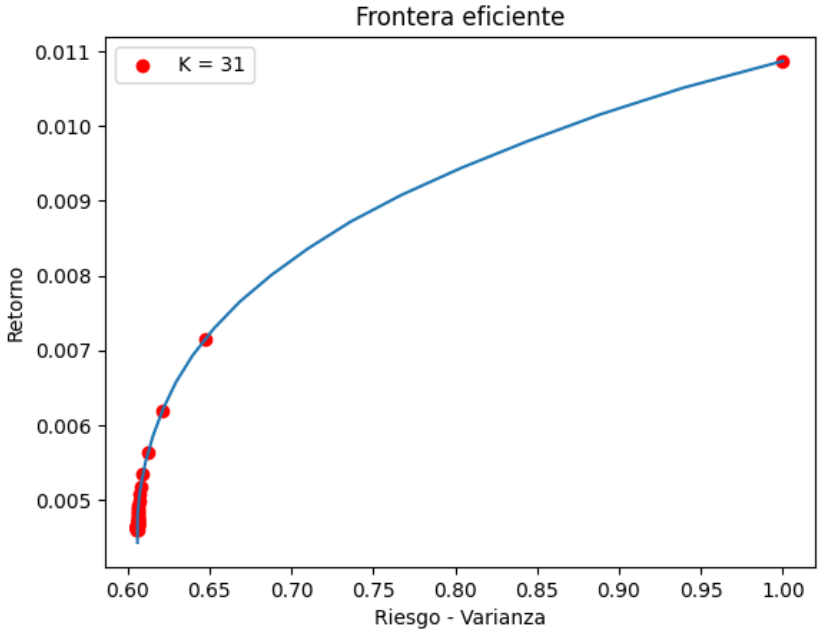

Figura 4: Frontera eficiente para $N = 31$