Optimización de Portafolios de Inversión con Restricción de Cardinalidad en Espacios Grandes de Acciones

por

Antonio Ortiz Ambriz

Tesis

Presentada el Programa de Graduados de Mecatrónica y Tecnologías de Información

Como requisito parcial para obtener el grado académico de

Maestro en Ciencias

con especialidad en

Sistemas Inteligentes

Tecnológico de Monterrey

Campus Monterrey

Agosto 2010

Reconocimientos

A Jaime Ortiz, Patricia Ambriz y Manuel Valenzuela.

Antonio Ortiz Ambriz

Tecnológico de Monterrey
Agosto 2010

Optimización de Portafolios de Inversión con Restricción de Cardinalidad en Espacios Grandes de Acciones

Antonio Ortiz Ambriz, M.C.
Tecnológico de Monterrey, 2010

Asesor de Tesis: Dr. Manuel Valenzuela Rendón

Este trabajo propone una representación basada en índices que tiene como propósito resolver el problema de optimizar portafolios de inversión de acuerdo a la formulación de Markowitz utilizando métodos evolutivos. Se busca encontrar portafolios óptimos a partir de espacios grandes de acciones, de tamaños que se encuentren en el mismo orden de magnitud que mercados completos. El problema se desea resolver con cardinalidad restringida, y la representación propuesta maneja esta restricción de forma natural.

La representación basada en índices consiste en optimizar un vector que identifica aquellos activos que se deben incluir en el portafolio, así como el peso que se les debe dar. Para encontrar cuales son los activos que logran minimizar el riesgo y maximizar el rendimiento se crea una lista ordenada que los incluya a todos, y se busca optimizar un vector de índices que hace referencia a esta lista. Adyacente a cada índice, se agrega un valor que representa el peso que el activo debe de tener en el portafolio.

La representación basada en índices se comparó con algoritmos utilizando una representación directa que consiste simplemente en un vector de pesos, en donde la restricción de cardinalidad se maneja utilizando sólo aquellos activos que tengan un mayor peso en el portafolio.

Se realizaron pruebas con varios parámetros de aversión al riesgo, en espacios de 100, 417, y 1000 acciones. Se reportan curvas de mejor encontrado contra evaluaciones de la función objetivo, diagramas de riesgo contra rendimiento de los mejores individuos encontrados por cada algoritmo, y diagramas de caja de las soluciones finales. Se observó que los algoritmos evolutivos con una representación basada en índices obtienen mejores resultados, en menos evaluaciones de la función objetivo que los algoritmos con representación directa.

Índice general

Capítulo 1

Introducción

Existen 1835 acciones en el *New York Stock Exchange* (NYSE) y 2744 acciones en el índice NASDAQ *Composite*. Si se quisiera tener un portafolio óptimo dentro de todo el mercado de acciones, se tendrían entonces que seguir 4579 acciones, sólo en los mercados de Estados Unidos, sin incluir bonos, futuros, o acciones en mercados extranjeros. Son estas grandes cantidades de acciones las que motivan el presente trabajo. Se busca diseñar un algoritmo que pueda encontrar portafolios en la frontera de eficiencia a partir de grandes cantidades de activos.

El método utilizado más comúnmente para optimizar portafolios es programación cuadrática. Sin embargo, la restricción de cardinalidad es una restricción no lineal, y el algoritmo de programación cuadrática no puede manejarla. Cuando se manejan miles de acciones, la restricción de cardinalidad se vuelve particularmente importante. Si el algoritmo utilizado no cuenta con un mecanismo para fijar el número de acciones en los portafolios que genera, todos estos serán portafolios de miles de acciones, lo que en la práctica causará que los costos de transacción se disparen, y que la inversión sea muy difícil de administrar.

Este trabajo propone una representación basada en índices que maneja la restricción de cardinalidad en forma natural, y permite a un algoritmo evolutivo resolver el problema en menor tiempo y con mejores resultados que otras heurísticas de búsqueda que manejan la restricción haciendo una reinterpretación de los portafolios encontrados. La representación basada en índices consiste en separar el problema de optimización de portafolios en dos subproblemas: el problema de cuales acciones, de todas las que se encuentran en el espacio, se deben comprar; y el de cuanto, en términos de porcentaje del portafolio, se debe comprar de cada una.

Se utilizaron dos algoritmos diferentes con la representación basada en índices. El primero es un algoritmo genético, que se encarga de optimizar todo el cromosoma y así resolver ambos subproblemas. El otro es un algoritmo memético, en el cual se busca que la parte evolutiva del algoritmo resuelva que acciones deben entrar en el portafolio, y la heurística de aprendizaje resuelve cuanto de cada acción debe comprarse para una combinación específica de acciones.

El trabajo, en resumen tiene la siguiente estructura. Se ofrece un capítulo con

antecedentes que se divide en dos secciones, administración de inversiones y algoritmos genéticos. La sección de administración de inversiones da una introducción al concepto de acciones. Describe la forma en que se estima el precio futuro de las acciones, muestra como se agrupan las inversiones en portafolios de inversión, y justifica el uso de estos portafolios, como método para reducir el riesgo. Se describen los métodos y los criterios utilizados para optimizar portafolios, y se habla de la restricción de cardinalidad.

La siguiente sección introduce al lector a los métodos de optimización ciega, particularmente a los algoritmos genéticos. Como ejemplo y estructura básica se describe el algoritmo genético simple, sus parámetros y sus variaciones más frecuentes. Se habla de los algoritmos meméticos, y luego se describen algunos trabajos que en el pasado han intentado resolver el problema de optimización de portafolios con métodos evolutivos.

El capítulo de metodología consiste en describir el trabajo que se realizó. Se utilizó un enfoque directo, que consiste en intentar resolver el problema utilizando métodos de búsqueda local de la forma más intuitiva posible. Se describe dicho enfoque, y se dan los detalles del algoritmo memético utilizado, así como funciones objetivo y la función de vecindad de los buscadores locales. Por último se muestran los resultados obtenidos en cuanto a calidad de soluciones, y tiempo de ejecución de los algoritmos.

Capítulo 2

Antecedentes

Este capítulo se divide en dos secciones. La sección de Administración de Inversiones da una introducción al concepto de instrumentos de inversión, principalmente acciones. Se describe la forma en que se estiman los valores futuros de una acción, y luego se introduce el concepto de portafolio de inversión. Se generaliza la obtención de los estimadores a portafolios, y se muestra cómo se puede diversificar una inversión para reducir el riesgo. Se muestran métodos para obtener portafolios óptimos, y se introduce el término de frontera de eficiencia, para terminar describiendo el problema de optimización de portafolios con restricción de cardinalidad.

La segunda sección consiste en familiarizar al lector con los métodos de optimización ciega, en particular con los algoritmos genéticos. Se comienza por describir de forma general los algoritmos genéticos, y definir algunos de los términos utilizados más frecuentemente. Después se describe el algoritmo genético simple, que consiste en un algoritmo genético en lo más esencial. Se describen algunas variaciones, que si bien hacen que un algoritmo genético deje de ser simple, se acercan más al algoritmo propuesto en este trabajo. Así mismo se describen algunos parámetros y los criterios que se deben utilizar para elegir los valores de estos. La última parte de esta sección está más orientada a este trabajo en particular. Se describen los algoritmos meméticos y algunos trabajos que han utilizado métodos de optimización evolutiva para resolver el problema de optimización de portafolios.

2.1. Administración de Inversiones

Una inversión consiste en obtener un pago a cambio de postergar el consumo. Al pago que se obtiene se le llama el *rendimiento* de una inversión. Existen diversos tipos de instrumentos de inversión, que pueden consistir en propiedades, préstamos, acciones, futuros, entre otros. Este trabajo hablará principalmente de acciones, pero el análisis puede generalizarse a cualquier instrumento para el cual se pueda obtener un estimador del rendimiento futuro y un estimador del riesgo.

Una acción es un instrumento que representan un porcentaje de una compañía.

Cuando una compañía necesita dinero para expandirse, emite acciones al mercado. El público puede entonces comprar estas acciones y se vuelve dueño parcial de esta compañía. Un inversionista que compra acciones espera que en el futuro sucedan una de dos cosas: la compañía obtiene ganancias, las cuales se deben repartir equitativamente a los dueños de acciones, o el valor de la compañía aumenta junto con el precio de la acción, en cuyo caso, el inversionista puede vender la acción a un precio mayor y obtener como rendimiento la diferencia entre el precio anterior y el precio actual.

La mayoría de las inversiones vienen acompañadas de un *riesgo*. En el caso de una acción, el riesgo es que el precio de la acción baje en vez de subir, en cuyo caso, el inversionista deberá vender la acción a un precio menor, y perder parte de su capital [12]. Un *inversionista racional* es aquel que prefiere acciones con mayor rendimiento antes que aquellas con menor rendimiento, y acciones con menor riesgo antes que acciones con mayor riesgo.

Formalmente, si se compra una acción en un tiempo $t-1$, y se vende en un tiempo t, se puede definir el rendimiento de una acción de acuerdo a la ecuación

$$r(t) = \frac{s(t) - s(t-1)}{s(t-1)} = \frac{s(t)}{s(t-1)} - 1 \tag{2.1}$$

en donde $s(t)$ es el precio de la acción en el tiempo t. El rendimiento se obtiene como un porcentaje porque el rendimiento real de una inversión depende del capital inicial.

2.1.1. Estimadores

Si un inversionista quiere tomar un decisión acerca de una inversión, debe poder estimar el rendimiento que obtendrá en el futuro, y debe estimar el riesgo de no obtenerlo. De acuerdo a Eugene Fama [5], la serie de tiempo del precio de una acción se comporta de forma similar a un paseo aleatorio. Si se supone que los cambios en el precio de una acción tienen una distribución normal, entonces el mejor indicador del rendimiento futuro que se puede obtener es el promedio de los rendimientos pasados. Para una acción i, definiremos entonces:

$$\bar{r}_i \equiv \mathrm{E}\left[r_i(t)\right] = \frac{1}{T}\sum_t r_i(t) \tag{2.2}$$

como el valor esperado del rendimiento.

El riesgo de una inversión representa la probabilidad de que el rendimiento sea diferente del esperado. Volviendo a la suposición de normalidad de los rendimientos, una forma de cuantificar esta probabilidad es obteniendo el promedio de las desviaciones del valor esperado de rendimiento a través del tiempo. Para estimar el riesgo de una

inversión se puede utilizar la desviación estándar de los rendimientos en el pasado.

$$\bar{\sigma}_i = \sqrt{\mathrm{E}\left[(r_i(t) - \bar{r}_i)^2\right]} \tag{2.3}$$

2.1.2. Portafolios de Inversion

Si se cuenta con un capital de inversión, una forma de reducir el riesgo es diversificar, que significa dividir el capital e invertirlo en más de una acción. Al diversificar se pueden reducir o eliminar las variaciones correspondientes a las acciones individuales, y obtener inversiones más estables. Como un ejemplo de diversificación supongamos que una inversión está compuesta por cantidades iguales de dos acciones de industrias diferentes. Si cada una de estas industrias se beneficia de la baja en los precios de la otra, entonces si una acción produce rendimientos positivos, la otra producirá rendimientos negativos. Como la mitad del capital se encuentra en cada una de las acciones, los rendimientos obtenidos se cancelarán. Se espera sin embargo que a largo plazo, ambas acciones tengan un rendimiento positivo, y es este rendimiento el que la inversión produce. A un capital que está dividido en varias inversiones se le llama un portafolio de inversión.

Un portafolio de inversión que cuenta con N acciones se puede representar como un vector de pesos

$$\mathbf{w} \equiv [w_1, w_2, ..., w_N] \tag{2.4}$$

Generalmente se normaliza el vector de pesos, de tal forma que

$$\sum_i w_i = 1 \tag{2.5}$$

De esta forma, cada peso del vector representa el porcentaje del capital invertido en esa acción en particular, y no una cantidad especifica.

Estrictamente hablando, los elementos del vector de pesos pueden tomar valores menores que cero. Cuando el peso de una acción en un portafolio es menor que cero, se dice que esta acción está *vendida en corto*. El termino *venta en corto* quiere decir que se pide prestada una cantidad determinada de acciones de una empresa para venderlas, con la esperanza de que el precio de las acciones baje. Una vez que el precio baje, se compran de nuevo esas acciones a un precio menor y se devuelven al propietario. De esta forma se saca provecho de los retornos negativos producidos por una acción.

Las leyes que regulan las ventas en corto no siempre permiten que modelarlas sea tan fácil como simplemente permitir pesos negativos. Es por esto que frecuentemente se restringen portafolios para incluir solamente pesos positivos. Es decir, el vector de

pesos debe cumplir la condición de que:

$$\forall i \;\; w_i \in [0,1] \tag{2.6}$$

Rendimiento Esperado de un Portafolio

Para cada acción i en un portafolio se puede obtener un valor esperado de rendimiento r_i. De esta forma se obtiene un vector de rendimientos

$$\mathbf{r} \equiv [\bar{r}_1, \bar{r}_2, ..., \bar{r}_N] \tag{2.7}$$

En cada momento, el rendimiento de un portafolio es la suma de sus rendimientos, es decir:

$$r_p(t) = \sum_i w_i r_i(t) \tag{2.8}$$

Usando este resultado, se puede ver que el valor esperado del rendimiento de un portafolio $\bar{r}_p$ es

$$\bar{r}_p = \mathrm{E}\left[r_p(t)\right] = \mathrm{E}\left[\sum_i w_i r_i(t)\right] \tag{2.9}$$

como el valor de los pesos del portafolio es constante en el tiempo, entonces:

$$\bar{r}_p = \sum_i w_i \mathrm{E}\left[r_i(t)\right] = \sum_i w_i \bar{r}_i \tag{2.10}$$

que se puede escribir también como la ecuación matricial

$$\boxed{\bar{r}_p = \mathbf{w}\mathbf{r}'} \tag{2.11}$$

Riesgo Esperado de un Portafolio

Se puede derivar el riesgo esperado de un portafolio a partir de la varianza de la serie de tiempo de rendimientos.

$$\bar{\sigma}_p^2 = \mathrm{E}\left[(r_p(t) - \bar{r}_p)^2\right] = \mathrm{E}\left[\left(\sum_{i=1}^{N} w_i r_i(t) - \sum_{i=1}^{N} w_i \bar{r}_i\right)^2\right] \tag{2.12}$$

$$\bar{\sigma}_p^2 = \mathrm{E}\left[\left(\sum_{i=1}^{N} w_i \left(r_i(t) - \bar{r}_i\right)\right)^2\right] \tag{2.13}$$

$$\bar{\sigma}_p^2 = \sum_{i=1}^{N}\sum_{j=1}^{N} w_i w_j \mathrm{E}\left[\left(r_i(t) - \bar{r}_i\right)\left(r_j(t) - \bar{r}_j\right)\right] \tag{2.14}$$

Ya que la covarianza de dos conjuntos diferentes de datos esta definida por

$$\sigma_{ij} = \mathrm{E}\left[\left(r_i(t) - \bar{r}_i\right)\left(r_j(t) - \bar{r}_j\right)\right] \tag{2.15}$$

Se pueden combinar las últimas dos ecuaciones para obtener

$$\bar{\sigma}_p^2 = \sum_{i=1}^{N}\sum_{j=1}^{N} w_i w_j \sigma_{ij} \tag{2.16}$$

Si se define una matriz de covarianzas de la forma

$$\mathbb{C} = \begin{bmatrix} \sigma_{11} & \sigma_{1,2} & \cdots & \sigma_{1N} \\ \sigma_{21} & \sigma_{22} & & \sigma_{2N} \\ \vdots & & \ddots & \vdots \\ \sigma_{N1} & \sigma_{N2} & \cdots & \sigma_{NN} \end{bmatrix} \tag{2.17}$$

Entonces la ecuación 2.16 se puede escribir como:

$$\boxed{\bar{\sigma}_p^2 = \mathbf{w}\mathbb{C}\mathbf{w}'} \tag{2.18}$$

Por Qué Diversificar

Una forma más formal de ilustrar la reducción del riesgo que se obtiene de la diversificación es considerar por un momento un portafolio en que todas las acciones participan con el mismo peso. La ecuación 2.16 entonces se vuelve

$$\bar{\sigma}_p^2 = \sum_{i=1}^{N}\sum_{j=1}^{N} \frac{1}{N^2} \sigma_{ij} \tag{2.19}$$

Si sumamos por separado las varianzas y las covarianzas tenemos

$$\bar{\sigma}_p^2 = \sum_{i=1}^{N} \frac{1}{N^2}\sigma_i^2 + \sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N} \frac{1}{N^2}\sigma_{ij} \tag{2.20}$$

$$\bar{\sigma}_p^2 = \frac{1}{N}\left[\sum_{i=1}^{N} \frac{1}{N}\sigma_i^2\right] + \frac{N-1}{N}\left[\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N} \frac{1}{N(N-1)}\sigma_{ij}\right] \tag{2.21}$$

En la última ecuación se puede ver que los términos entre corchetes corresponden a el promedio de las varianzas $\langle\sigma_i^2\rangle$, y al promedio de las covarianzas $\langle\sigma_{ij}\rangle$, respectivamente. Se puede expresar esta ultima ecuación entonces como

$$\bar{\sigma}_p^2 = \frac{1}{N}\left\langle\sigma_i^2\right\rangle + \frac{N-1}{N}\left\langle\sigma_{ij}\right\rangle \tag{2.22}$$

o factorizando y agrupando términos

$$\bar{\sigma}_p^2 = \frac{1}{N}\left[\left\langle\sigma_i^2\right\rangle - \left\langle\sigma_{ij}\right\rangle\right] + \left\langle\sigma_{ij}\right\rangle \tag{2.23}$$

La ecuación 2.23 se divide entonces en dos partes, una que es proporcional a $\frac{1}{N}$ y otra que no. Cuando N crece, la primera parte de la ecuación se acerca a cero, dejando sólo la segunda parte. Ya que que la varianza de las acciones individuales sólo aparece en esta parte, se puede eliminar la contribución de las varianzas individuales al riesgo de portafolio si se aumenta el número de acciones. Cuando N crece entonces, la varianza del portafolio sólo depende del promedio de las covarianzas entre sus componentes [4].

2.1.3. Optimización de Portafolios de Inversión

Hasta ahora se han definido estimadores para predecir el riesgo y el rendimiento de una acción, y se generalizaron estos estimadores a portafolios de acciones lo que llevó a la conclusión que diversificando una inversión se puede reducir el riesgo de la misma. A esta conclusión se llegó suponiendo que se compra la misma cantidad de todas las acciones en el espacio. Sin embargo el modelo de un portafolio que se describió permite asignar a cada acción un peso diferente y existe, para un conjunto de acciones dado, un conjunto de pesos que minimiza el estimador de riesgo.

Hay que recordar que un inversionista racional siempre elegirá inversiones de menor riesgo antes que acciones de mayor riesgo. Sin embargo, también elegirá acciones de mayor rendimiento antes que acciones de menor rendimiento. Si existieran dos portafolios A y B, en donde el portafolio A tiene menor riesgo que el portafolio B, y el rendimiento de A es mayor que el de B, un inversionista racional siempre elegiría el portafolio A. Por otro lado, si el portafolio A tiene menor riesgo que el portafolio B,

pero el portafolio B tiene mayor rendimiento que el portafolio A, entonces no existe un portafolio óptimo que cualquier inversionista deba preferir, sino que depende de cuanto le disguste el riesgo al inversionista, y de cuanto le guste un aumento en el rendimiento. El problema de optimizar un portafolio se vuelve entonces un problema intrínsecamente multiobjetivo.

La cantidad de riesgo que un inversionista está dispuesto a aceptar a cambio de un mayor rendimiento es diferente para cada inversionista y suele ser una característica difícil de definir. Por esta razón, no existe tal cosa como un portafolio óptimo (o un par de portafolios A y B), sino una frontera óptima en el sentido de Pareto de posibles inversiones a la cual se le llama *frontera de eficiencia* y se muestra en la figura 2.1.

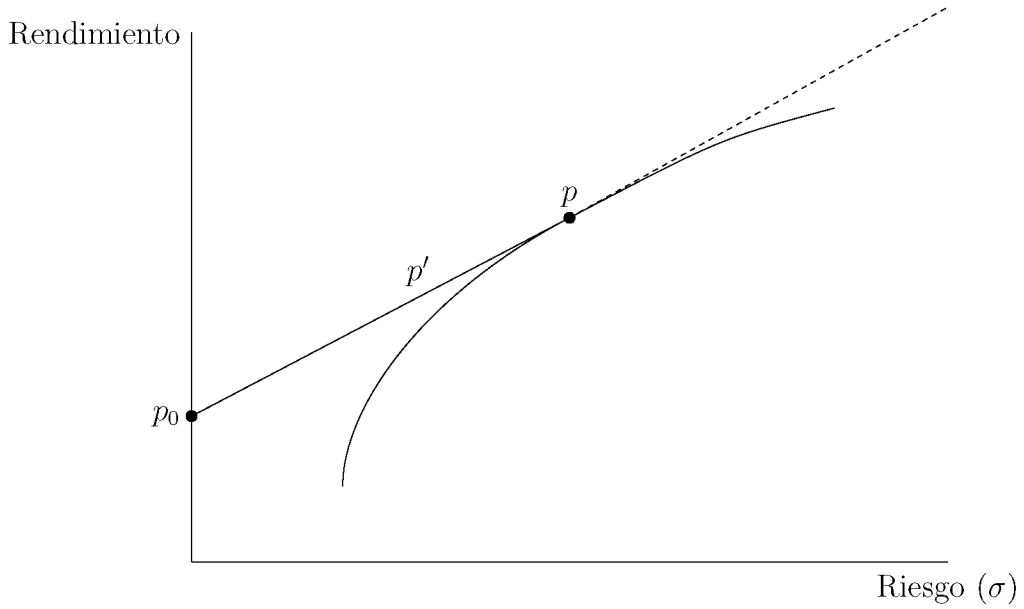

Figura 2.1: Frontera de eficiencia en una gráfica de Rendimiento contra Riesgo. El punto p es el portafolio de tangencia, el punto p_0 es la acción de riesgo cero. La línea recta $\overline{pp_0}$ muestra las combinaciones lineales entre el portafolio de tangencia y la acción de riesgo cero. La linea punteada son combinaciones válidas si se cuenta con la posibilidad de obtener un préstamo con una tasa de interés igual al rendimiento de p_0.

Optimización Utilizando Programación Cuadrática

El problema de optimizar un portafolio se puede plantear como el problema de encontrar un vector $\mathbf{w}$ tal que:

$$\min_{\mathbf{w}} [\mathbf{w}(\mathbb{C})\mathbf{w}'] \tag{2.24}$$

Sujeto a:

$$\mathbf{r}'\mathbf{w} = r \tag{2.25}$$

$$\sum_i w_i = 1 \tag{2.26}$$

$$0 \leq w_i \leq 1 \quad \forall i \tag{2.27}$$

Se puede observar que en la fórmula 2.24 lo que se está minimizando es el riesgo del portafolio, y el rendimiento se restringe a un valor deseado r en la restricción 2.25. Si se quisiera obtener toda la frontera de eficiencia de acuerdo a este método se puede resolver el problema para varios valores del rendimiento deseado. Para obtener el rango de valores posibles de rendimiento se puede utilizar como límite inferior la solución al problema sin incluir la restricción 2.25, y como límite superior el rendimiento de la acción con mayor rendimiento de las que se encuentran disponibles. El problema tal como se plantea se puede resolver por el método de programación cuadrática [4, 8].

Otra opción para convertir el problema multiobjetivo en un problema de un sólo objetivo es optimizar una combinación lineal de ambos objetivos. Si se pretende maximizar el rendimiento pero minimizar el rendimiento, se puede utilizar una función de la forma

$$\max_{\mathbf{w}} \left[\alpha \bar{r}_p - \beta \bar{\sigma}_p^2\right] \tag{2.28}$$

En donde ambos parámetros α y β son mayores que cero. En este caso, α es una medida de cuanto le gusta el aumento en el rendimiento esperado al inversionista, y β es una medida de cuanto le disgusta el aumento en el riesgo. En la práctica es más fácil tener un sólo parámetro, así que se pueden unificar α y β en un parámetro λ de aversión al riesgo tal que $\lambda \in [0, 1]$, $\alpha = (1 - \lambda)$ y $\beta = \lambda$ La función entonces se vuelve

$$\max_{\mathbf{w}} \left[(1 - \lambda)\bar{r}_p - \lambda \bar{\sigma}_p^2\right] \tag{2.29}$$

Es fácil observar que el parámetro λ, definido de esta forma permite encontrar toda la frontera de eficiencia. Si se fija que $\lambda = 0$, entonces $\alpha = 1$ y $\beta = 0$; es decir, al inversionista no le disgusta el aumento en el riesgo, y sólo le importa el rendimiento, lo cual produciría el portafolio de máximo riesgo en toda la frontera. Por otro lado si $\lambda = 1$,

entonces $\beta = 1$ y $\alpha = 0$, y al inversionista no le interesa aumentar el rendimiento, sino sólo encontrar el portafolio de mínimo riesgo.

Al sustituir las ecuaciones 2.11 y 2.18 se obtiene el problema de optimización:

$$\max_{\mathbf{w}} \left[(1-\lambda)\,\mathbf{r}'\mathbf{w} - \mathbf{w}(\lambda\mathbb{C})\mathbf{w}'\right] \tag{2.30}$$

Sujeto a:

$$\sum_i w_i = 1 \tag{2.31}$$

$$0 \leq w_i \leq 1 \quad \forall i \tag{2.32}$$

La solución a este problema es un portafolio diferente para cada valor de λ. La frontera eficiente se puede obtener resolviendo la función de optimización 2.30 para varios valores de λ.

Relación de Sharpe

Existen en el mercado acciones que garantizan un rendimiento, tales como los bonos gubernamentales. A estas aciones se les llama acciones de riesgo cero y su varianza, así como su covarianza con cualquier otra acción del mercado es cero. El riesgo y el rendimiento de cualquier combinación de una inversión, con la acción de riesgo cero cae sobre la línea recta que une a estos dos activos.

Si los porcentajes de la combinación se encuentran ambos en el intervalo $[0, 1]$, entonces el riesgo y el rendimiento caen en el segmento de recta entre la acción de riesgo cero y el portafolio riesgoso. Si el peso del portafolio en la combinación cumple $x_p > 1$, entonces $x_{rf} < 0$, por que se debe cumplir que $x_p + x_{rf} = 1$, y la combinación cae en un punto de la línea recta más allá del portafolio: esta región se muestra en la figura 2.1 como una linea punteada. Esta combinación es posible si se puede pedir un préstamo con una tasa de interés igual al rendimiento de la acción de riesgo cero.

Al valor de la pendiente de esta línea recta se le llama *Relación de Sharpe* y esta dado por:

$$S = \frac{r_p - r_{rf}}{\sigma_p} \tag{2.33}$$

donde r_{rf} es el rendimiento de la acción de riesgo cero. Es fácil ver que el portafolio que maximiza la Relación de Sharpe debe encontrarse en la frontera de eficiencia 2.1. A este portafolio se le llama el portafolio de tangencia. Se puede ver también en la figura 2.1 que cualquier portafolio que se encuentre en la frontera de eficiencia y que tenga un riesgo menor que el del portafolio de tangencia, puede ser mejorado con una combinación del portafolio de tangencia y la acción de riesgo cero. Si se cuenta con la posibilidad de obtener préstamos con una tasa de interés igual a r_{rf}, entonces también

cualquier portafolio sobre la frontera de eficiencia, con un riesgo mayor al de la acción de riesgo cero se puede mejorar. De esta forma, la nueva frontera de eficiencia consiste siempre en una combinación del portafolio de tangencia con la acción de riesgo cero, y sólo se requiere encontrar este portafolio de tangencia para encontrar toda la frontera eficiente.

La relación de Sharpe se muestra para ilustrar otro tipo de formas de optimizar portafolios. En este trabajo se buscará la frontera de eficiencia tal como se muestra el la función 2.30, sin considerar la existencia de un activo libre de riesgo. Así mismo, a pesar de que cuando se permite al inversionista realizar ventas en corto, se puede resolver el problema de forma analítica, existen buenos motivos para resolver el problema sin permitir ventas en corto Este trabajo se concentrará en optimizar portafolios de inversión en que todos los pesos son mayores que cero.

2.1.4. Portafolios de Cardinalidad Restringida

La restricción de cardinalidad consiste en limitar el número de acciones que entran en el portafolio. Ya que no es posible expresar esta restricción como una función lineal del vector de pesos, no es posible utilizar el método de programación cuadrática para encontrar portafolios óptimos con números definidos de acciones. Sin embargo existen buenas razones por las cuales encontrar portafolios de cardinalidad restringida, especialmente si se buscan portafolios en espacios grandes de acciones.

Si se quiere invertir en un portafolio, se debe identificar un conjunto óptimo de pesos para las acciones con las cuales se cuenta y luego se deben comprar las acciones de acuerdo a este conjunto de pesos, al precio actual. Después de un cierto tiempo, el precio de estas acciones probablemente haya cambiado, y con el, el capital. Este cambio generalmente significa que el porcentaje del capital invertido en cada acción ya no es aquel que se identificó como un conjunto óptimo de pesos. Es en este caso cuando hay que *rebalancear* un portafolio. Esto significa que se deben realizar transacciones entre los activos en los que se ha invertido, para volver a tener el conjunto óptimo de pesos.

Si se compra un portafolio, y la intención es conservarlo durante un periodo largo, este requerirá ser rebalanceado periodicamente. El costo de este rebalanceo aumenta conforme aumenta el número de acciones en el portafolio, pues cada una de las transacciones que son necesarias tiene un costo. Esta es una de las razones por las cuales es preferible tener pocas acciones antes que muchas. Ademas del rebalanceo, el esfuerzo de administrar y monitorear un portafolio y los costos de transacción en que se incurre cuando se quiere actualizar el portafolio de acuerdo a nueva información, son algunos de los argumentos que motivan la optimización de portafolios con restricción de cardinalidad.

Volviendo a la ecuación 2.23, que muestra el riesgo para un portafolio de N acciones con igual peso, se puede observar de nuevo que se compone de un término que se puede

eliminar diversificando, y un término que siempre se encontrará en el portafolio, sin importar cuantas acciones se incluyan. Si en el portafolio se incluyera sólo una acción, el riesgo esperado sería $\langle \sigma_i^2 \rangle$. Sin embargo, por más acciones que se incluyan, el riesgo siempre será mayor que $\langle \sigma_{ij} \rangle$. Para que el riesgo esperado esté $\alpha \langle \sigma_i^2 \rangle$ por encima el mínimo riesgo, el tamaño del portafolio está dado por:

$$N = \frac{1}{\alpha} \frac{\langle \sigma_i^2 \rangle}{[\langle \sigma_i^2 \rangle - \langle \sigma_{ij} \rangle]} \tag{2.34}$$

Se puede definir la variable γ como el porcentaje del riesgo del portafolio de una sola acción que se puede eliminar diversificando, de forma que:

$$N = \frac{1}{\alpha\gamma} \tag{2.35}$$

Elton [4] reporta valores de γ que se encuentran entre 0.893 y 0.56. Lo que quiere decir que si se quiere que el riesgo del portafolio exceda al mínimo riesgo por el 10 % del riesgo de una sola acción, entonces se necesitan entre 11 y 18 acciones, dependiendo del valor de γ del mercado. Se puede observar entonces que el número de acciones que se necesita para eliminar la mayor parte del riesgo es pequeño. Para las siguientes secciones de este trabajo se buscarán portafolios de cardinalidad restringida a 15 acciones, cuando se aplique esta restricción.

Hasta ahora se ha definido métodos para estimar los resultados de una inversión, tanto para una acción como para un portafolio de acciones. Se usaron estos estimadores para construir un modelo que permitiera obtener portafolios que disminuyeran efectivamente el riesgo de las inversiones. En lo que resta del capítulo se describirán los algoritmos que se utilizan en este trabajo, tanto como propuesta como comparación. También se muestran algunos trabajos que se han realizado para resolver el problema de optimización de portafolios, tanto con restricción de cardinalidad como sin restricciones, utilizando las heurísticas de búsqueda descritas.

Este trabajo se enfocará en resolver el problema de optimización de portafolios con restricción de cardinalidad en espacios grandes de acciones. Los portafolios que se buscarán son de 15 acciones, pues se pudo en esta sección que no son necesarios más activos para eliminar gran parte del riesgo. La función que deben optimizar los portafolios buscados es la mostrada en 2.30, en donde se establece una combinación lineal del riesgo y el rendimiento, con un parametro λ de aversión al riesgo.

2.2. Algoritmos Genéticos

Esta sección da una introducción a los *algoritmos genéticos*. Se presenta primero una definición general del AG como método de búsqueda ciega y optimización. Se define el algoritmo genético simple, se describen algunas de las decisiones que hay que tomar en el diseño de un algoritmo genético como son el criterio de parado, y el tamaño de población. Se describen algunas de las variaciones al algoritmo genético simple que se usaron en este trabajo, incluyendo los algoritmos meméticos, que son el método de solución propuesto para el problema de optimización de portafolios en mercados grandes. Por último se describen un par de trabajos realizados en el pasado que utilizan algoritmos evolutivos para resolver instancias similares del problema.

Los AGs son una técnica de búsqueda que está inspirada en el mecanismo de evolución natural. Se basa esencialmente en utilizar la supervivencia del más fuerte para explorar el espacio en búsqueda de soluciones óptimas, y en utilizar cruce y mutación para generar soluciones nuevas a partir de pedazos de soluciones que han funcionado bien anteriormente [6].

Se espera que para un problema en particular, se cuente con una función a optimizar. A esta función se le llamará *función objetivo*. Al conjunto $\mathbb{X}$ de soluciones posibles de la función objetivo se le llamará *espacio de búsqueda*. Una posible solución del problema se expresa como x donde $x \in \mathbb{X}$. Formalmente, la *función objetivo* $f(x)$ se define como $f : \mathbb{X} \to \mathbb{Y}$, donde $\mathbb{Y}$ debe ser un conjunto ordenado. Si se tienen dos soluciones x_i y x_j, se dice que x_i es mejor solución que x_j si y sólo si $f(x_i) > f(x_j)$.

Para utilizar un AG, no es necesario conocer la estructura de la función objetivo, es por esto que se dice que el AG es un algoritmo de *Optimización Ciega*. En general, tampoco es necesario que el rango de la función objetivo sea una calificación absoluta, siempre que sea un conjunto ordenado.

El AG es un método de búsqueda poblacional, lo que significa que en cada paso contiene una colección de soluciones independientes entre sí. A esta colección se le llamará *población*, y a cada elemento de esta población se le llama un *individuo*. Un individuo es una posible solución $x \in \mathbb{X}$. La evaluación de la función objetivo para cada individuo se llama *aptitud*.

En un AG cada individuo es una cadena de caracteres. Es recomendable, pero no necesario, que los individuos sean cadenas en un alfabeto de cardinalidad baja. Una *representación* es la transformación del espacio de estas cadenas, al espacio de búsqueda. La elección de una representación es lo más importante para armar un AG que funcione eficientemente. Con una representación apropiada, los AGs son sumamente robustos, pues no asumen que la función a optimizar es continua y derivable, o que tiene sólo un máximo local.

La idea básica detrás de un AG es que la función objetivo puede ser una caja negra, de la cual no se conoce la estructura o el mecanismo según el cual entrega

una calificación. Si se quiere optimizar una función objetivo así, es necesario evaluar varios intentos, y guardar el que mejor resultado obtenga. El AG funciona como una forma de elegir que intentos se evalúan, de acuerdo a los intentos que se ha encontrado que producen buenos resultados. Cada individuo generado por el AG es uno de estos intentos.

2.2.1. El Algoritmo Genético Simple

El algoritmo genético simple (AGS) cuenta con tres operadores básicos:

- Reproducción
- Cruce
- Mutación

Algoritmo 1 Pseudocódigo del Algoritmo Genético Simple

```
P ← GeneraPoblación
P ← EvalúaPoblación(P, f(x))
for all Generaciones do
  P ← Reproducción(P)
  P ← Cruce(P)
  P ← Mutación(P)
  P ← EvalúaPoblación(P, f(x))
end for
```

El pseudocódigo del AGS se muestra en la figura 1. El algoritmo comienza por generar una población de cadenas aleatorias. Cada una de estas cadenas es un individuo, que representa un intento de encontrar una solución que optimice la función objetivo. Antes de comenzar el ciclo principal del algoritmo, se evalúan los individuos. Estos dos pasos constituyen la inicialización del AG. A partir de aquí comienza el ciclo principal, y cada vuelta del ciclo se le llama una *generación*. Una generación del AG consiste en asignar copias a los individuos más aptos de la población mediante el operador de *reproducción*. De esta manera se eliminan aquellas soluciones cuya evaluación es baja y se favorecen aquellas que mayor evaluación tengan. Los operadores de *cruce* y *mutación* tienen como propósito generar nuevos individuos a partir de las piezas que componen los individuos en la población actual. Ya que se asignaron más copias a los individuos más aptos, los nuevos individuos se generarán a partir de estas piezas. El AG entonces trata de optimizar la función objetivo, juntando aquellas piezas de solución que producen individuos mejor evaluados. Al final de la generación, cada individuo se vuelve a evaluar, y se regresa al inicio del ciclo principal. A continuación se describen los operadores básicos del AG.

El AGS es generacional. En un AG generacional, cada generación se destruye toda la población y se sustituye con los descendientes. En un AG no generacional, la población no se destruye totalmente, y sólo una parte de los individuos se sustituye o se concatena a la población actual.

El operador de reproducción asigna copias a los individuos de acuerdo a su aptitud y sustituye a la población o a un conjunto de ésta por una nueva población. Un AG simple utiliza selección proporcional, que significa que el valor esperado del número de copias que se asignan a un individuo, es proporcional a su aptitud. Para implementar selección proporcional, se necesita que la función objetivo arroje una calificación absoluta para cada individuo. Dicho formalmente, si n_t el número de copias que un individuo tiene en la población en la generación t, $f(x)$ su aptitud, y $\langle f \rangle$ el promedio de la aptitud de todos los individuos que actualmente se encuentran en la población, un método de reproducción puede llamarse selección proporcional si el número esperado de copias en la población después de la reproducción es:

$$\langle n \rangle_{t+1} = n_t \frac{f(x)}{\langle f \rangle} \tag{2.36}$$

Frecuentemente, y especialmente cuando se quiere utilizar selección proporcional, la operación de reproducción se implementa como una ruleta. Se asigna a cada individuo un espacio proporcional a su aptitud en la rueda de la ruleta, que se gira tantas veces como individuos se quieran generar. Por cada giro de la rueda de ruleta, el individuo ganador tiene una copia en la nueva población. La ruleta es un método de reproducción ilustrativo, pero ineficaz, y se recomienda utilizar otros métodos en la práctica.

El operador de cruce de un punto elige al azar dos cadenas, y un punto intermedio en ellas. Los caracteres que siguen al punto elegido se intercambian, y quedan dos nuevas cadenas. Esta operación se efectúa sobre todos los pares de la población, elegidos aleatoriamente. Por último se aplica el operador de mutación, que consiste en cambiar ligeramente uno de los números en algunas cadenas.

2.2.2. Ejemplo del Funcionamiento de un Algoritmo Genético Simple

Un ejemplo ilustrativo del funcionamiento de un AGS está dado por Goldberg [6]. Se pretende optimizar la función objetivo $f(x) = x^2$ en el rango $[0, 31]$. Para codificar el espacio, los individuos utilizados serán cadenas binarias de cinco bits, lo que significa que

$$0 \leftarrow [00000]$$
$$31 \leftarrow [11111]$$

Se selecciona una población aleatoria de cuatro individuos

$$\begin{array}{cccccc} 0 & 1 & 1 & 0 & 1 \leftarrow & 13 \\ 1 & 1 & 0 & 0 & 0 \leftarrow & 24 \\ 0 & 1 & 0 & 0 & 0 \leftarrow & 8 \\ 1 & 0 & 0 & 1 & 1 \leftarrow & 19 \end{array}$$

Al evaluarla, se obtienen las siguientes aptitudes

$$\begin{array}{rcccccl} f(0 & 1 & 1 & 0 & 1) = & 13^2 = & 169 \\ f(1 & 1 & 0 & 0 & 0) = & 24^2 = & 576 \\ f(0 & 1 & 0 & 0 & 0) = & 8^2 = & 64 \\ f(1 & 0 & 0 & 1 & 1) = & 19^2 = & 361 \end{array}$$

La el promedio de la evaluación para este caso es 293, y como cada individuo tiene sólo una copia, el número esperado de copias para cada individuo, dado por la ecuación 2.36, es

$$\begin{array}{ccccc} 0 & 1 & 1 & 0 & 1 \leftarrow 0.58 \\ 1 & 1 & 0 & 0 & 0 \leftarrow 1.97 \\ 0 & 1 & 0 & 0 & 0 \leftarrow 0.22 \\ 1 & 0 & 0 & 1 & 1 \leftarrow 1.23 \end{array}$$

Al aplicar el operador de reproducción, se obtiene la siguiente población:

$$\begin{array}{ccccc} 0 & 1 & 1 & 0| & 1 \leftarrow 13 \\ 1 & 1 & 0 & 0| & 0 \leftarrow 24 \\ 1 & 1 & 0| & 0 & 0 \leftarrow 24 \\ 1 & 0 & 0| & 1 & 1 \leftarrow 19 \end{array}$$

En esta nueva población se puede observar el efecto que se espera del operador de reproducción. Aquellos individuos con una alta aptitud (en este caso el $11000 \leftarrow 24$) recibió dos copias en la nueva población, mientras que el individuo con menos aptitud ($01000 \leftarrow 8$) no recibió ninguna.

Se aplica el operador de cruce en los puntos indicados, y se obtiene la nueva población, que al ser evaluada, se obtiene

$$\begin{array}{rcccccl} f(0 & 1 & 1 & 0 & 0) = & 12^2 = & 144 \\ f(1 & 1 & 0 & 0 & 1) = & 25^2 = & 625 \\ f(1 & 1 & 0 & 1 & 1) = & 27^2 = & 729 \\ f(1 & 0 & 0 & 0 & 0) = & 16^2 = & 256 \end{array}$$

En este caso, el promedio de las evaluaciones aumenta de 293 a 439. En este punto, se vuelve a aplicar el operador de reproducción y se continúa con el resto del algoritmo.

Para este ejemplo es suficiente mostrar una generación.

Lo que es importante notar de este ejemplo, es como son aquellas soluciones que descienden de las mejores soluciones las que producen mejores resultados. Se puede ver como en la población inicial el individuo con mejor evaluación es $11000 \leftarrow 24$. A este individuo se le otorgaron dos copias en la siguiente población. Fueron estas dos copias las que al cruzarse con los siguientes mejores individuos encontraron las soluciones mejor evaluadas[1].

2.2.3. Parámetros de un Algoritmo Genético

Existen varios parámetros que es necesario elegir para afinar apropiadamente un algoritmo genético. Los más evidentes son la probabilidad de cruce (P_c) y probabilidad de mutación (P_m). Normalmente se usa una probabilidad de cruce alta ($P_c \approx 1$) y una probabilidad de mutación baja ($P_m \approx 0.1$).

El criterio según el cual un AG se debe detener y reportar una solución es uno de los puntos menos triviales en el diseño de un algoritmo eficiente. Un algoritmo que se deje corriendo seguirá produciendo nuevas soluciones hasta que la evaluación del individuo más fuerte sea suficientemente fuerte para que las copias que genera llenen la población. Cuando esto sucede, ya no hay en el AG diversidad suficiente para producir soluciones nuevas. El operador de mutación puede a veces generar nuevos individuos suficientemente fuertes para competir con el individuo más fuerte. Sin embargo la probabilidad de que esto ocurra es baja, y la mayoría de las veces sólo se generan individuos que son rápidamente eliminados por las copias del individuo que actualmente domina la población. Si la solución encontrada cuando el AG ya no puede encontrar nuevas soluciones es suficientemente buena, el criterio de parado debería detener el algoritmo, de tal forma que no se pierda tiempo evaluando una y otra vez soluciones iguales. Sin embargo, si esta solución no es suficientemente buena, entonces se dice que el algoritmo llegó a *convergencia prematura.*

La opción más simple es parar el AG después de un número determinado de generaciones. La ventaja de parar el algoritmo así es que se tiene una idea muy acertada de cuánto tiempo tomará una corrida. Sin embargo, se debe elegir cuidadosamente la cantidad de generaciones en las que se detendrá, pues para cada problema, la función de crecimiento de la evaluación del mejor individuo encontrado por el AG suele ser diferente. Cuando se detiene el algoritmo de acuerdo a un número de generaciones, se corre el riesgo tanto de que se siga explorando el espacio y se sigan obteniendo buenas soluciones, como de que el algoritmo haya convergido mucho tiempo atrás y todo el tiempo restante ha sido desperdiciado.

Otra opción para detener un AG es obtener alguna medida de convergencia, y establecer un valor límite. La forma más sencilla de medir la convergencia de una

[1]Este ejemplo fue tomado del primer capítulo del libro de Goldberg [6] de manera textual

población es la desviación estándar de la evaluación de los individuos. A este tipo de convergencia se le llama convergencia en aptitud.

El tamaño de población usado en un algoritmo genético debería ser suficiente para que exista una alta probabilidad de encontrar la mayoría de los pedazos de la solución óptima en la población inicial. Cuando se resuelven problemas que requieren cromosomas muy grandes, se requieren poblaciones muy grandes. Esto suele causar que el algoritmo tome mucho tiempo. El riesgo con utilizar poblaciones más pequeñas es que el algoritmo forme la mejor solución que se puede encontrar con las piezas que se encuentran en la población inicial, pero esta solución se encuentre debajo del óptimo y el algoritmo converja prematuramente.

2.2.4. Variaciones al Algoritmo Genético Simple

Selección por Torneo Otro método de implementar reproducción es *selección por torneo*. Para implementar selección por torneo no es necesaria una calificación, sólo que el rango de la función objetivo sea un conjunto ordenado. Para implementar selección por torneo, se define un tamaño m, se ordenan los individuos aleatoriamente, y se hace competir a cada individuo con sus m vecinos más cercanos. Por cada competencia ganada, se asigna una copia al individuo en la nueva población.

Inyecciones de Diversidad Usualmente, en especial cuando la probabilidad de mutación es baja, es difícil que un algoritmo que converge prematuramente encuentre nuevas soluciones. Incluso si el algoritmo no se detiene por varias generaciones más, seguirá encontrando las mismas combinaciones que encontraba antes. Para evitar la convergencia prematura, se utilizan inyecciones de diversidad, que consisten en eliminar a los peores individuos en la población y remplazarlos por individuos generados aleatoriamente. De esta forma se introducen nuevas piezas a la población, que el AG puede utilizar para encontrar mejores soluciones. Para decidir en qué momento se quiere aplicar inyecciones de diversidad, se puede elegir un número definido de generaciones, o se puede utilizar algunos de los criterios de convergencia.

Algoritmo Genético de Genes Virtuales (vgGA) En [15] se propone una forma alternativa de implementar un AG tradicional binario. Los operadores de cruce y mutación se pueden traducir de un alfabeto a otro a partir de operaciones de módulo. De esta forma se pueden aplicar estos operadores sobre los individuos en su representación natural, y obtener el mismo resultado que si se aplicaran sobre los individuos en un alfabeto de cardinalidad baja. Esta implementación permite generalizar el AG tradicional a alfabetos de cualquier cardinalidad.

Mapeo Lineal El AGS está definido en un alfabeto de baja cardinalidad. De acuerdo al teorema de esquemas [6] el alfabeto que más cantidad de esquemas contiene por bit de información es el alfabeto binario, lo que lo hace el más conveniente para codificar los parámetros de la función objetivo de un AG. El problema es que si se utiliza un alfabeto binario, el rango de estos parámetros siempre estará en $\mathbb{Z} \in [0, 2^l - 1]$, en donde l es el número de bits utilizados. Si se desean codificar parámetros que se encuentren en un espacio diferente, se puede usar *mapeo lineal* para establecer una función que transforme del espacio de los enteros en el rango $[0, 2^l - 1]$ al espacio de los reales en el rango $[x_{\text{mín}}, x_{\text{máx}}]$.

Si se tiene un segmento de cromosoma b tal que $b \in \mathbb{Z} \wedge b \in [0, 2^l - 1]$, y se quiere evaluar un parámetro x tal que $x \in \mathbb{R} \wedge x \in [x_{\text{mín}}, x_{\text{máx}}]$ se utiliza la función

$$x(b) = \left(\frac{x_{\text{máx}} - x_{\text{mín}}}{2^l - 1} \right) b + x_{\text{mín}} \tag{2.37}$$

La resolución que se obtenga sobre el parámetro x dependerá del número de bits l que se elija usar.

2.2.5. Uso de Algoritmos Genéticos para Resolver el Problema de Optimización de Portafolios

Varias trabajos han utilizado representaciones diferentes para resolver el problema de optimización de portafolios usando algoritmos genéticos, o algoritmos híbridos.

Entre los primeros trabajos se encuentra el de Chang[2], que compara el comportamiento de tres heurísticas diferentes para resolver el problema de optimización de portafolios con cardinalidad restringida. La representación utilizada en este trabajo es la misma que la usada por Chang. Se explicará a detalle en el capítulo 3, pero consiste básicamente en un vector de índices que hace referencia a acciones en una lista ordenada, y un vector de pesos que indica qué peso se le debe de dar a esa acción en el portafolio. Se utilizó una codificación real, de forma que bajo el operador de cruce, todas las acciones que se encuentran en el nuevo individuo, se encontraban en alguno de los padres. Así mismo, los pesos de las acciones estaban adyacentes a los índices, de forma que todas las acciones que pasaran al nuevo individuo, tendrían el mismo peso que en tenían en el padre del que se heredaron. Para que los individuos se mantuvieran cumpliendo con las restricciones, se utilizó un operador de reparación que reinterpreta a los individuos, pero no modifica los cromosomas.

Las heurísticas que se compararon fueron un AG, búsqueda tabú y recocido simulado. Se utilizó un algoritmo genético de estado estable, con una población de 100 individuos y cruce uniforme. Todos los nuevos individuos eran modificados por el operador de mutación, que consistía en multiplicar por 1.1 o por 0.9 un elemento del vector

de pesos elegido aleatoriamente. Este operador de mutación se utilizó tambien en el algoritmo de búsqueda tabú y en recocido simulado como función de vecindad. En el algoritmo de recocido simulado se redujo la temperatura cada $2N$ evaluaciones, donde N es el número total de acciones utilizadas, y con un valor de $\alpha = 0.95$ Para los tres algoritmos se llevaron a cabo $1000N$ evaluaciones de la función objetivo.

Las pruebas se llevaron a cabo en cinco conjuntos de datos provenientes de cinco índices. Cada conjunto de datos consiste en sólo aquellas acciones que cuentan con información de precios semanales entre marzo de 1992 y septiembre de 1997. Los conjuntos resultantes fueron 31 acciones del índice *Hang Seng*, 85 acciones del *DAX*, 89 acciones del *FTSE*, 98 acciones del *S&P100* y 225 acciones del *Nikkei*. Para probar los algoritmos sobre el problema de cardinalidad restringida, se buscaron portafolios que consistieran de exactamente 10 acciones.

Chang utilizó los tres algoritmos para encontrar la frontera de eficiencia sin restricción de cardinalidad. Esta frontera de eficiencia se comparó con la encontrada con el método de programación cuadrática. Se encontró que la heurística que más se acercaba a la frontera de eficiencia era el AG. Luego se probó el comportamiento de los tres algoritmos al encontrar la frontera de eficiencia con la restricción de cardinalidad. En este caso no se comparó contra el resultado encontrado por programación cuadrática, sino que se creó una frontera de eficiencia de puntos no dominados entre los puntos encontrados. Se pudo ver que aunque el AG es la heurística que más puntos contribuyó a esta frontera de eficiencia, los tres algoritmos aportan un número de puntos muy similar. No se muestran resultados que digan a cuál región de la frontera de eficiencia contribuye más cada uno de los algoritmos.

Carlos Martínez[10] probó diversos métodos de optimización multiobjetivo para encontrar la frontera de eficiencia con diversas restricciones. Se utilizan cinco conjuntos de acciones provenientes de la Bolsa Mexicana de Valores, con 10, 15, 20, 25 y 50 acciones. Se utilizó primero un AG generacional con función objetivo 2.30, con tres valores del parámetro de aversión al riesgo y una representación que consiste simplemente en un vector de pesos. Esta representación se describe más detalladamente en el capítulo 3 y se le llama *enfoque directo*. Este algoritmo se comparó con programación cuadrática, pero el AG no resulta competitivo si las restricciones son lineales. Se probó también una heurística para resolver la restricción de cardinalidad, que consiste en encontrar un portafolio óptimo con programación cuadrática, utilizar sólo aquellas k acciones de mayor rendimiento, y renormalizar el portafolio. El resultado indica que el AG que evalúa la restricción de cardinalidad encuentra portafolios con mejor evaluación que la heurística descrita. Luego se probaron diferentes métodos de optimización multiobjetivo para resolver el problema con cardinalidad restringida, y se encontraron buenos resultados sobre un problema de 50 acciones, con 250×10^3 evaluaciones de la función objetivo.

Aranha [1], aprovechó que se puede ver un portafolio como una combinación de

subportafolios, y usó una representación de árbol binario. Cada nodo del árbol tenía un valor, y representaba un activo. Este activo podía ser una acción real, en el caso de nodos terminales, o podía ser un subportafolio en el que los nodos hijo eran los activos, y el porcentaje de participación estaba dado por el valor del nodo. Se implementaron operadores especiales de cruce, que consistían en intercambiar ramas enteras. El punto en el cual se genera el cruce se obtenía a partir de la evaluación de los subárboles. Esta representación mostró ser útil para eliminar del portafolio acciones cuya participación no es significativa.

2.3. Algoritmos Meméticos

2.3.1. Descripción de los Algoritmos Meméticos

Los *algoritmos meméticos* (AM) son algoritmos híbridos entre AGs y alguna forma de heurística local, que se utilizan como método de optimización ciega. Este método consiste en una población de individuos, tal como la de un AG, en la cual, al ser evaluado, cada individuo es sometido a algún tipo de aprendizaje de acuerdo con la heurística local que se esté utilizando. El resto del algoritmo funciona de la misma manera que un AG, con los mismos operadores de cruce, mutación y reproducción [9].

La forma en que se da este aprendizaje es sencilla. Suponiendo por ejemplo, que se utiliza recocido simulado como heurística local, se debe generar una población de individuos con las mismas características que los de un AG, pero en el momento de evaluarlos, en vez de hacerlo directamente en la función objetivo, se utilizan estos individuos como puntos iniciales de un algoritmo de recocido simulado. La aptitud de cada individuo es la evaluación del mejor punto encontrado por el algoritmo de recocido simulado, lo que significa que el individuo más fuerte no es aquel que maximice la función objetivo, sino aquel que tenga una mayor capacidad de mejorar dentro de la heurística local.

Existen en este punto dos opciones: el individuo puede conservar su forma original y sólo su evaluación es modificada por la heurística local, o el nuevo individuo mejorado encontrado por la heurística local puede sustituir al individuo original. A la primera opción se le llama *efecto Baldwin* y a la segunda *Lamarckismo* [16]. En cierto sentido, un AM con efecto Baldwin se puede ver como un AG común y corriente, en que la función objetivo contiene una heurística local.

La motivación que lleva a implementar un AM es que el mecanismo evolutivo cuente con un operador adicional, en la forma de esta heurística local, que asista a los operadores del AG en su búsqueda de soluciones. Es decir, que la heurística local debe mejorar los individuos para hacer más fácil al AG el encontrar el óptimo.

La figura 2 muestra el pseudocódigo de un AM. La única diferencia con el AG en la figura 1 es la sustitución de el operador *Evaluación* por el operador *Aprendizaje*.

Algoritmo 2 Pseudocódigo del Algoritmo Memético

```
P ← GeneraPoblación
P ← Aprendizaje(P, Opt(f(x)))
for all Generaciones do
    P ← Reproducción(P)
    P ← Cruce(P)
    P ← Mutación(P)
    P ← Aprendizaje(P, Opt(f(x)))
end for
```

Dentro de los argumentos al operador, debe ir la función objetivo, pero también la heurística local. Si la población original se modifica o si sólo se modifica su evaluación depende de si se utiliza un AM con Lamarckismo o con efecto Baldwin.

2.3.2. Trabajos Previos en Algoritmos Meméticos y El Problema de Optimización de Portafolios

Darrel Whitley [16] lleva a cabo una comparación entre un AM con Lamackismo y uno con efecto Baldwin. Se modela de forma analítica el comportamiento de ambos algoritmos y de un AG simple al intentar resolver un problema engañoso, y se prueban experimentalmente el desempeño de los tres algoritmos al resolver problemas numéricos de optimización. Se concluye que para ciertos problemas un AM con efecto Baldwin puede tener un mejor desempeño que el de un AM con Lamarckismo. Se observó que en algunos de los problemas probados, el AM con efecto Baldwin evita con más facilidad los óptimos locales, mientras que los otros dos algoritmos convergían a soluciones subóptimas.

Streichert [13] usó una representación que intentaba aprovechar el hecho de que los portafolios frecuentemente no incluyen todas las acciones del mercado. Para esto adaptó un mecanismo para que el AG pudiera eliminar acciones facilmente del portafolio.

Se evaluó el uso de algoritmos meméticos para resolver el problema de optimización de portafolios como un problema de optimización multiobjetivo. Se utilizó un algoritmo genético multiobjetivo, con una población de 500 individuos y se guardaban 250 puntos no dominados con una distancia de compartición de $\sigma = 0.01$. Se utilizó un conjunto de 31 acciones, y se buscaron portafolios con cardinalidad restringida a 2, 4, y 6 acciones, y sin restricción de cardinalidad. La heurística local utilizada fue un mecanismo de reparación que convertía en portafolios factibles aquellos portafolios que violaban alguna restricción. El algoritmo memético utilizó este mecanismo de reparación como función de aprendizaje de efecto Baldwin, y de Lamarckismo.

Se comparó el comportamiento de dos diferentes representaciones. La primera fue

una representación directa que consistía en un vector de pesos **w**, que contiene un elemento w_i por cada acción en el espacio. En la otra representación, la primera parte del cromosoma era un vector de pesos **w** como el de la representación directa, pero se le agregaba una segunda parte que consistía en un vector de decisión **B**, en que cada elemento era un número binario B_i que decidía si la acción entraba al portafolio o no. Se hicieron pruebas utilizando efecto Baldwin y Lamarckismo, en donde la función de aprendizaje consistía en un mecanismo de reparación que convertía portafolios que violaban alguna restricción en portafolios factibles.

Se concluyó que la representación extendida con el vector de decisión **B** obtiene mejores resultados que la representación directa. Así mismo, se pudo ver que ambas representaciones se benefician fuertemente del uso de Lamarckismo sobre usar simplemente el efecto Baldwin.

En un artículo posterior, Streichert [14] propuso una nueva representación que consiste en separar el vector de decisión del vector de pesos, y aplicar los operadores de cruce y mutación independientemente. Se realizaron pruebas con los mismos parámetros que en [13] y se probó que la nueva representación es más eficaz que la representación directa, y que ambas representaciones obtienen mejores resultados utilizando un AM con Lamarckismo.

2.4. Recocido Simulado

Otro método de optimización ciega es *recocido simulado*. Este algoritmo es un método de búsqueda local que es una generalización del método de búsqueda de alpinista que permite aceptar peores soluciones con el propósito de escapar de óptimos locales.

La búsqueda de alpinista opera sobre un punto del espacio de búsqueda a la vez. En cada paso del algoritmo se genera un punto vecino de acuerdo a una *función de vecindad* y se evalúa. Si la evaluación del vecino es mayor que la del punto actual, el vecino se toma como el nuevo punto actual. Si no, el vecino se descarta y se busca un nuevo vecino. Este método está garantizado incompleto, pues si el algoritmo llega a un óptimo local, no hay forma en que pueda salir de él [11].

Recocido simulado tiene como objetivo arreglar este defecto permitiendo movimientos que alejen del óptimo. Esta basado en el *algoritmo de Metropolis*, que es un método para simular el proceso de formación de cristales mediante una disminución gradual de la temperatura. A esta forma de generar cristales se le llama *Recocido*

2.4.1. Funcionamiento

En un sistema termodinámico formado por una colección de partículas a una baja temperatura, estas tienden a ordenarse en la configuración de menor energía. Si la tem-

peratura no es cero, existe la probabilidad de que las partículas se encuentren en una configuración que tenga una energía mayor. Esta probabilidad es mayor conforme se aumenta la temperatura, pero disminuye entre mayor sea la diferencia entre la configuración de menor energía y la nueva configuración. Cuando se forman cristales por medio de un proceso de recocido, se espera que en el inicio del proceso, las partículas se ordenen con mayor probabilidad en configuraciones de baja energía, pero que se permitan movimientos entre estas partículas que permitan elevar la energía del sistema, de forma que en los siguientes movimientos se puedan encontrar configuraciones de energía aún menor. Conforme se reduce la temperatura, estos movimientos se van haciendo menos probables, de forma que el sistema empieza a reducir su energía a cada movimiento, sin permitir que vuelva a subir [7].

El método de recocido simulado, al igual que el de búsqueda de alpinista, comienza con un punto inicial, y requiere de una función vecino. Cada iteración del algoritmo se genera un vecino. Si $f(\mathbf{x}_t)$ es la evaluación del punto actual $\mathbf{x}_t$, y $f(\mathbf{x'}_t)$ es la evaluación del punto vecino $\mathbf{x'}_t$, el nuevo punto $\mathbf{x}_{t+1}$ está dado por la ecuación

$$\mathbf{x}_{t+1} = \begin{cases} \mathbf{x}'_t & \textbf{si } f(\mathbf{x'}_t) \geq f(\mathbf{x}_t) \\ \chi & \textbf{si } f(\mathbf{x'}_t) < f(\mathbf{x}_t) \end{cases} \tag{2.38}$$

En donde χ se elige de acuerdo a la temperatura T según

$$\chi = \left[\mathbf{x}'_t : P\left(\exp\left(\frac{f(\mathbf{x}_t) - f(\mathbf{x'}_t)}{T} \right) \right), \mathbf{x}_t : P\left(1 - \exp\left(\frac{f(\mathbf{x}_t) - f(\mathbf{x'}_t)}{T} \right) \right) \right] \tag{2.39}$$

2.4.2. Programa de Enfriamiento

La mayor complicación al diseñar un algoritmo de recocido simulado es establecer una *programa de enfriamiento*. El programa de enfriamiento se define como un mapeo del espacio de las iteraciones al espacio de la temperatura. Es decir, a que temperatura deberá llevarse a cabo cada intento de solución evaluado por el algoritmo. Usualmente se elige una temperatura inicial, que debe definirse pensando en el rango de valores de la función objetivo, y se va disminuyendo de acuerdo a

$$T_{n+1} = \alpha T_n \tag{2.40}$$

donde $\alpha \in [0, 1)$. Generalmente el valor de α es grande (> 0.9) de forma que la temperatura no se vuelva insignificante demasiado rápido. Es importante notar que los subíndices de las temperaturas usados en la ecuación 2.40 son n y no t para indicar que no se reduce la temperatura a cada iteración.

Un ejemplo de un programa de enfriamiento comúnmente utilizada se encuentra en [7], en donde se utiliza recocido simulado para encontrar configuraciones de circuitos integrados. Esta agenda de temperaturas cuenta con dos ciclos principales anidados.

Se empieza con una temperatura inicial, y se hacen iteraciones a una temperatura constante hasta que se aceptan L puntos, o se han evaluado M puntos. Es entonces que se modifica la temperatura, de acuerdo a la ecuación 2.40 y se vuelve a entrar a este ciclo. Una vez que han pasado C ciclos sin mejorar el valor de la función objetivo, el sistema se considera congelado, y se detiene el algoritmo. El algoritmo 3 muestra el algoritmo de recocido simulado con este programa de enfriamiento.

En el artículo de Chang [2] se utiliza recocido simulado para resolver el problema de optimización de portafolios. La agenda de temperaturas es similar a la mostrada en [7] y en el algoritmo 3, pero en este caso se lleva a cabo una cantidad definida de iteraciones. Se efectúan $2N$ iteraciones, en donde N es el número de acciones en el espacio, a una temperatura constante antes de reducirla según la ecuación 2.40. Este ciclo se llevó a cabo 500 veces, de forma que el número de evaluaciones de la función objetivo fuera $1000N$ y coincidiera con los otros algoritmos que se estaban probando. La temperatura inicial utilizada es $\frac{1}{10}f(\mathbf{x}_0)$, y se fijó $\alpha = 0.95$.

Algoritmo 3 Pseudocódigo del algoritmo de recocido simulado

$T \leftarrow$Temperatura Inicial
Ciclos Sin Mejora $\leftarrow 0$
Punto $\leftarrow$ *PuntoAleatorioInicial*
repeat
  Intentos Aceptados $\leftarrow 0$
  PuntoInicial $\leftarrow$ Punto
  repeat
    Vecino $\leftarrow$ *FuncionVecindad*(Punto, Tamaño de Vecindad)
    if *FuncionObjetivo*(Punto) $<$ *FuncionObjetivo*(Vecino) **then**
      if *NumeroAleatorio*$[0,1)$ $<$ $\exp\left(\frac{FuncionObjetivo(\text{Punto}) - FuncionObjetivo(\text{Vecino})}{T}\right)$ **then**
        Punto $\leftarrow$ Vecino
        Intentos Aceptados $\leftarrow$ Intentos Aceptados+1
      end if
    else
      Punto $\leftarrow$ Vecino
      Intentos Aceptados $\leftarrow$ Intentos Aceptados+1
    end if
  until (Intentos Aceptados $\geq L$)$\vee$(Intentos $\geq M$)
  if *FuncionObjetivo*(Punto) $\leq$ *FuncionObjetivo*(PuntoInicial) **then**
    Ciclos Sin Mejora $\leftarrow$ Ciclos Sin Mejora +1
  end if
  $T \leftarrow \alpha T$
until Ciclos Sin Mejora $\geq C$

Capítulo 3

Metodología

En este capítulo se describen los experimentos realizados. Se probaron dos enfoques principales. Primero un *enfoque directo*, que consiste en utilizar varios algoritmos de búsqueda de una forma intuitiva. El enfoque directo se utilizó para encontrar portafolios óptimos bajo la restricción de cardinalidad y sin restricciones, en espacios de 100, 417, y 1000 acciones. Se describen los parámetros utilizados, la función objetivo y la función de vecindad utilizada por los buscadores locales.

El segundo enfoque utilizado tiene como propósito manejar naturalmente la restricción de cardinalidad, especialmente en espacios grandes de acciones. Consiste en un AG y un AM que utilizan una representación basada en índices. Al igual que para el enfoque directo, se muestran los parámetros utilizados para los experimentos, y la función objetivo.

3.1. Datos Utilizados y Definición del Problema

Los conjuntos de datos utilizados para probar los algoritmos se obtuvieron a partir de acciones del índice *Standard&Poor's 500* y del *NYA*. Se obtuvieron precios de cierre diarios de todas las acciones de ambos índices entre el 1 de enero y el 31 de diciembre del 2006. Aquellas acciones que tuvieran precios faltantes que no correspondieran a días inhabiles se eliminaron del conjunto. Las acciones resultantes fueron 417 acciones de *Standard&Poor's 500* y 1579 del *NYA*. Se utilizaron dos conjuntos del índice *Standard&Poor's 500* , uno consiste en las primeras 100 acciones del índice *Standard&Poor's 500* , y el otro de las 417 acciones, el tercer conjunto consiste de las primeras 1000 acciones del *NYA* tomadas en orden alfabético.

A partir de estas serie de tiempo, se obtuvieron rendimientos diarios según la ecuación 2.1

$$r(t) = \frac{s(t) - s(t-1)}{s(t-1)} = \frac{s(t)}{s(t-1)} - 1$$

Se obtuvo el vector de rendimientos esperados con elementos definidos por la ecuación

2.2

$$\bar{r}_i \equiv \mathrm{E}\left[r_i(t)\right] = \frac{1}{T}\sum_t r_i(t)$$

así como la matriz de covarianzas definida en 2.17, cuyo elemento ij es:

$$\sigma_{ij} = \mathrm{E}\left[\left(r_i(t) - \bar{r}_i\right)\left(r_j(t) - \bar{r}_j\right)\right]$$

Tanto el vector de rendimientos, como la matriz de covarianzas se anualizaron de acuerdo a las ecuaciones mostradas en el libro de Edwin Elton [4]

$$\bar{r}_{\text{anual}} = \left(1 + \bar{r}_{\text{diario}}\right)^{252} - 1 \tag{3.1}$$

$$\sigma_{ij} = \sigma_{ij} \times 252 \tag{3.2}$$

A partir de estos vectores de rendimiento y matrices de covarianzas , dado un valor del parámetro λ de aversión al riesgo, se busca optimizar la función objetivo 2.30

$$f(x, \lambda) = \left[(1 - \lambda)\mathbf{r}'\mathbf{w} - \mathbf{w}(\lambda\mathbb{C})\,\mathbf{w}'\right]$$

Tanto con la restricción de cardinalidad como sin ella.

Para probar el funcionamiento de los algoritmos en varias regiones de la frontera de eficiencia, se varía el valor de el parametro de aversión al riesgo λ. Se debe hacer énfasis en que todos los algoritmos utilizados son algoritmos de optimización de un sólo objetivo. El propósito de este trabajo no es barrer toda la frontera de eficiencia, sino encontrar un algoritmo que pueda encontrar un buen portafolio para un valor definido de λ. Si se quisiera obtener la frontera de eficiencia completa usando este algoritmo, se puede hacer corriéndolo para diferentes valores de λ elegidos de forma que produzcan puntos tan espaciados como se quiera. La elección de los valores correctos de λ para generar puntos igualmente espaciados no es un problemas trivial, y esta más allá del objetivo de este trabajo, pero probablemente varíe de acuerdo al conjunto de datos utilizado y probablemente sea mejor opción utilizar un algoritmo de optimización multiobjetivo como el utilizado en [10].

Para este trabajo se probó el algoritmo con 11 valores diferentes de λ en el rango $(0, 1]$ distribuidos según la función $f(x) = x^4$. Para comparar los algoritmos se fijó un límite de evaluaciones de la función objetivo de 50×10^3. De esta forma todos los algoritmos cuentan con la misma cantidad de oportunidades de obtener información de la función objetivo. En el capítulo 4 se comparan los resultados de estas corridas con los resultados de utilizar programación cuadrática. Así mismo se muestran curvas de mejor encontrado para poder observar la evolución de las soluciones conforme se permite al algoritmo hacer más evaluaciones de la función objetivo.

3.2. Solución Utilizando un Enfoque Directo

Se resolvió el problema usando un enfoque directo que consiste en aplicar un algoritmo de búsqueda de alpinista un algoritmo de recocido simulado y un AG. Se utilizó la representación más intuitiva para estos tres algoritmos, y se resolvió el problema. Se utilizó cada algoritmo para encontrar portafolios óptimos de cardinalidad definida y portafolios óptimos sin restricción de cardinalidad.

El enfoque directo no maneja la restricción de cardinalidad de forma natural. Para obtener portafolios con un número limitado de acciones, la función objetivo utiliza sólo aquellos portafolios que más peso tengan en el portafolio. De esta manera, la restricción de cardinalidad se incluye en la función objetivo, a pesar de que el algoritmo no se limite a puntos en el espacio de búsqueda con un número definido de acciones.

3.2.1. Representación

La representación utilizada por los algoritmos con enfoque directo es la misma que se usa en [10]. Para resolver el problema en un espacio de N acciones, se obtiene un vector $\mathbf{x}$ de N valores el rango $[0, 1)$. Estos valores se normalizan de acuerdo a

$$\mathbf{w} = \frac{1}{\sum_1^N x_i}\mathbf{x} \tag{3.3}$$

para obtener un vector de pesos, y se evalúan de acuerdo a la función objetivo 2.30.

Para obtener portafolios de cardinalidad restringida a k, antes de normalizar se toman los k valores más grandes, y el resto se hace cero. De esta forma el algoritmo modifica el portafolio para cumplir la restricción antes de ser evaluado. Este enfoque en la restricción de cardinalidad tiene la desventaja de que el algoritmo explora una gran cantidad de puntos redundantes, pues una parte del cromosoma no se está utilizando.

En el algoritmo genético se utilizaron cromosomas con N segmentos en los que cada segmento representaba el peso de una acción en el portafolio. Se codificaron los segmentos en un alfabeto binario y se utilizaron tres bits para representar cada uno de los pesos. De esta forma el valor de cada uno de los segmentos es un número entero en el rango $[0, 7]$ y se utilizó mapeo lineal para hacer que los pesos se encontraran en el rango $[0, 1)$.

3.2.2. Función de Vecindad

El algoritmo de recocido simulado requiere ademas una función vecino. Se utilizó una función que representa un intercambio de acciones. La función recibe como entrada un vector de pesos, y un tamaño de vecindad τ que se encuentra en el rango (0,1]. Se eligen dos acciones al azar en el portafolio; la primera es la vendedora y

la segunda es la compradora. El peso en el portafolio de la acción vendedora w_v se multiplica por $1 - \tau$, y al peso de la acción compradora se le suma $w_v\tau$. Esta función vecino tiene la ventaja de preservar la suma de un conjunto de pesos. Esto significa que si un portafolio se encuentra normalizado, esta normalización se conserva.

3.2.3. Parámetros

Ya que el AG tiene tres bits por segmento, la probabilidad de mutación por bit que se usó fue de $P_m = 1 - \left(1 - \frac{1}{4}\right)^{\frac{1}{3N}}$, de tal forma que, si cada individuo tiene N segmentos, la probabilidad de que un individuo sufra alguna mutación es de $\frac{1}{4}$. La probabilidad de cruce P_c que se usó para todos los AG es de 1. Se utilizó selección de torneo como método de selección. Para cada tamaño de problema, se utilizó como tamaño de población el número par más cercano a $\frac{N}{2}$.

Tanto con el algoritmo de de recocido simulado como el de búsqueda de alpinista se utilizó un tamaño de vecindad $\tau = \frac{1}{2}$. Para el algoritmo de recocido simulado, la temperatura inicial es igual a la evaluación del primer individuo, y se utilizó un valor de α tal que la temperatura se redujera en un 90 % cada 20 ciclos de temperatura constante. Se realizaron 500 ciclos de temperatura constante, cada uno de estos de 100 intentos. Con el algoritmo de búsqueda de alpinista se realizaron 50×10^3 evaluaciones de la función objetivo.

3.3. Solución Utilizando una Representación Basada en Índices

Se utilizó una representación similar a la utilizada en [2]. Con la representación utilizada se puede manejar la restricción de cardinalidad de forma natural. Esto se hace buscando no sólo los pesos de las acciones, sino cuales acciones deberán entrar en el portafolio. Se probaron dos algoritmos con esta representación: un AG y un AM. En ambos se busca que la parte evolutiva del algoritmo encuentre el conjunto de acciones que minimicen el riesgo y maximicen el rendimiento cuando se incluyen en un portafolio. La diferencia es que el AM cuenta con la ayuda de un algoritmo de búsqueda local para encontrar que peso se le debe de dar a cada acción para optimizar la función objetivo.

El operador de *MejoraPoblación* es una función que recibe el individuo, y modifica los segmentos del cromosoma correspondientes a los pesos, de tal forma que los pesos en el individuo resultante correspondan a un portafolio óptimo con ese conjunto de individuos. Para hacer esto se utiliza un algoritmo de búsqueda local, de la misma forma en que se implementan en el enfoque directo. Al mismo tiempo se debe permitir que el AM pase por suficientes generaciones como para que explore suficiente del espacio. El algoritmo de búsqueda local entonces debe ser un algoritmo rápido aunque aproximado.

Para agilizarlo, en lugar de utilizar un punto inicial aleatorio, se utiliza el vector de pesos actual.

3.3.1. Representación

Encontrar un portafolio óptimo en un espacio que incluya más de 500 acciones en el que se incluyan todas éstas no sólo es impráctico, sino que rara vez sucede. En general los portafolios que se encuentran en la frontera de eficiencia no incluyen todas las acciones del mercado, sino un conjunto reducido. Para reducir el espacio de búsqueda del portafolio óptimo sin dejar de observar ninguna acción se indexaron las acciones y se utilizó un vector de k índices, en donde cada elemento corresponde a una acción en el espacio. De esta forma se puede elegir cuantas acciones serán incluidas en el portafolio y el espacio de búsqueda sólo incluye aquellos portafolios que satisfacen la restricción de cardinalidad.

A cada elemento i del vector de índices, se le asignó un peso w_i. Cada peso w_i se colocó adyacente a su índice. De esta forma, cuando se cruzan los individuos, los índices tienden a cargar consigo el peso que tenían en el cromosoma padre. En la figura 3.1 se ilustra cómo se forman los cromosomas.

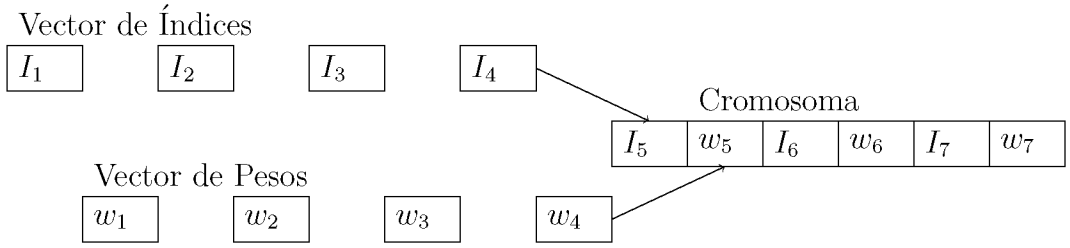

Figura 3.1: Diagrama de un cromosoma. Cada cromosoma se forma a partir de un vector de pesos y un vector de índices. Los elementos de los vectores se intercalan, para que bajo la operación de cruce se mantenga la proporción de cada acción en el individuo anterior

Dado que la función vecino descrita en los algoritmos de búsqueda local produce valores no enteros, se utilizó un mapeo lineal para convertir los valores de los enteros formados por los números binarios en el cromosoma, a números reales. De esta forma se aprovecha que el algoritmo preserva la normalización de los individuos.

3.3.2. Función Objetivo

En el AM la evaluación de cada individuo es aquella correspondiente al mejor individuo encontrado por el algoritmo de búsqueda local. El AG en cambio, evalúa directamente cada uno de los individuos sin utilizar el algoritmo de búsqueda local. La función objetivo utilizada para optimizar los pesos del portafolio recibe como parámetro

el cromosoma, el vector de rendimientos del espacio entero, la matriz de covarianzas, y un parámetro de aversión al riesgo λ. Separa el cromosoma en un vector de índices y un vector de pesos. Utiliza el vector de índices para obtener un subvector de rendimientos y una submatriz de covarianzas, y normaliza el vector de pesos de tal forma que

$$\sum_i w_i = 1 \tag{3.4}$$

Una vez que se tienen estos indicadores, se obtiene un estimador del riesgo y el rendimiento del portafolio de acuerdo a las ecuaciónes 2.11 y 2.16. La evaluación final del portafolio está dada por la ecuación 2.30

3.3.3. Parametros

Para ambos algoritmos basados en índices se utilizó una probabilidad de cruce $P_c = 1$ y una probabilidad de mutación $P_m = 0.01$. Se utilizaron tres bits para representar los pesos, tal como en el enfoque directo. Para los índices se usó el mínimo número de bits necesarios para representar a todas las acciones del mercado. Se utilizó un mapeo lineal para hacer que los segmentos correspondientes a los pesos se encontraran en el rango $[0, 1)$ y los segmentos correspondientes a los índices en el rango $[1, N]$. Se utilizó una población de 20, 50 y 80 para los problemas con 100, 417, y 1000 acciones, respectivamente. Todos los algoritmos se pararon a las 50×10^3 evaluaciones de la función objetivo. Se utilizó un algoritmo de búsqueda de alpinista, con tamaño de vecindad $\tau = \frac{1}{2}$ limitado a 10 iteraciones como heurística local del AM.

Capítulo 4

Resultados

En este capítulo se muestran los resultados de utilizar un enfoque directo y un enfoque basado en índices para encontrar portafolios óptimos en espacios de 100, 417, 1000 acciones. Se utilizó una combinación lineal de el riesgo y el rendimiento como función objetivo, con un parámetro λ de aversión al riesgo. Para observar el comportamiento de los algoritmos en diversos puntos de la frontera de eficiencia se utilizaron 11 valores del parámetro λ. Se comparan los desempeños de los diversos algoritmos, y se muestran también los resultados obtenidos contra los resultados obtenidos por un algoritmo de programación cuadrática.

4.1. Descripción del Análisis de Resultados

Ya que la escala de valores del riesgo y la del rendimiento son generalmente diferentes, el valor promedio de la evaluación de las soluciones encontradas en la parte superior de la frontera de eficiencia suele ser mayor que el de aquellas soluciones encontradas en la región de bajo riesgo.

En lugar de la evaluación dada por la ecuación 2.30 se utiliza una evaluación normailzada $\delta(x)$ que consiste en la diferencia porcentual entre la evaluación de un portafolio según la función objetivo utilizada y la evaluación del mejor individuo encontrado por un algoritmo de programación cuadrática. Es decir, si $x_{PQ(\lambda)}$ es la solución encontrada por programación cuadrática para un valor de λ, y $x_{A(\lambda)}$ es la solución encontrada por el algoritmo A, entonces la evaluación normalizada es

$$\delta(x_A) = \frac{f(x_{PQ(\lambda)}, \lambda) - f(x_{A(\lambda)}, \lambda)}{|f(x_{PQ(\lambda)}, \lambda)|} \tag{4.1}$$

De esta forma la evaluación de una solución toma valores similares para todos los valores de λ y la solución encontrada por un algoritmo es buena conforme δ se aproxime a 0

El comportamiento de los algoritmos se observa en cuatro tipos de figuras.

Curvas de Mejor Encontrado (CME) El propósito de las CME es el de mostrar la evolución de las soluciones encontradas por los algoritmos. Son gráficas de la evaluación de la mejor solución encontrada por cada algoritmo contra el número de evaluaciones de la función objetivo que se necesitaron para encontrar esa solución. Las CME mostradas promedian los valores de las soluciones encontradas por los algoritmos en cada una de las corridas, y para diferentes valores de λ.

Solucion Final Las gráficas de las soluciones finales son diagramas de caja que muestran la calidad promedio de la mejor solución encontrada por cada algoritmo, y la variación en estas soluciones. El propósito de estas gráficas es evaluar el desempeño final del algoritmo y la confiabilidad en las soluciones que produce. De manera similar que en las CME, se promedian los valores de δ para cada λ y se muestran los resultados de las diversas corridas.

Diagramas de Riesgo contra Rendimiento (Diagrama RR) La mejor solución encontrada por cada algoritmo se grafica en un diagrama de Riesgo contra Rendimiento. También se muestra en este diagrama la solución encontrada por programación cuadrática. El propósito de este diagrama es mostrar que tan cerca de la frontera de eficiencia están los puntos encontrados.

4.2. Resultados Obtenidos en un Espacio de 100 acciones

La figura 4.1 muestra la localización en un diagrama RR de los mejores portafolios encontrados por cada uno de los algoritmos a partir un espacio de 100 acciones. Los algoritmos con enfoque directo fueron utilizados sin restricción de cardinalidad. En este diagrama se puede ver que los pocos puntos fuera de la frontera encontrada por programación cuadática pertenecen todos al algoritmo de recocido simulado. El resto de los algoritmos, incluidos los basados en índices, hacen un buen trabajo en encontrar valores en la frontera de eficiencia.

Los resultados mostrados en la figura 4.2 vienen de agregar la restricción de cardinalidad a los algoritmos con enfoque directo. El algoritmo de búsqueda de alpinista sigue haciendo un buen trabajo, salvo por los puntos de mayor rendimiento que se quedan un poco abajo en la curva. Sin embargo el AG con enfoque directo deja de encontrar buenas soluciones para más allá de los dos primeros valores de λ.

Las figuras 4.3 y 4.4 muestran la evolución promedio de los algoritmos en la búsqueda de las soluciones mostradas en las figuras 4.1 y 4.2 respectivamente La figura 4.3 muestra la CME de los algoritmos con enfoque directo, sin restricciones, y de los algoritmos basados en índices. Recordando que según la corrección de la evaluación, el

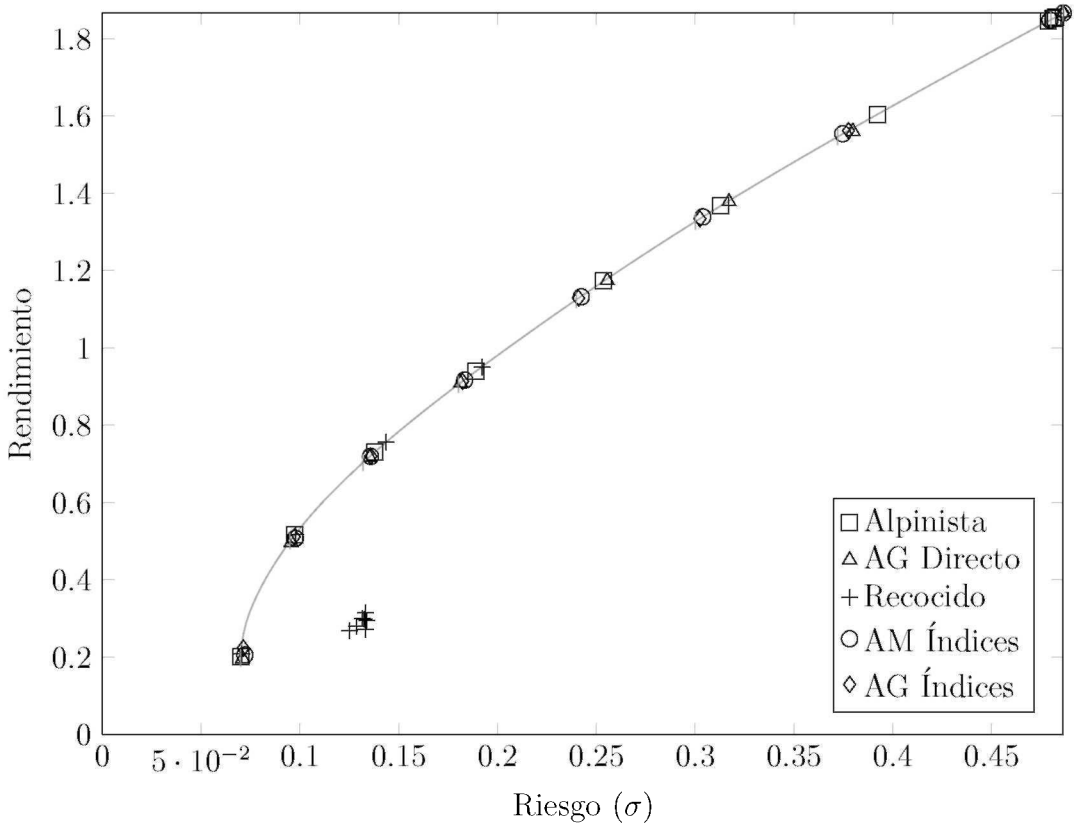

Figura 4.1: Puntos en el Diagrama RR encontrados por los algoritmos sin restricción de cardinalidad en un espacio de 100 acciones

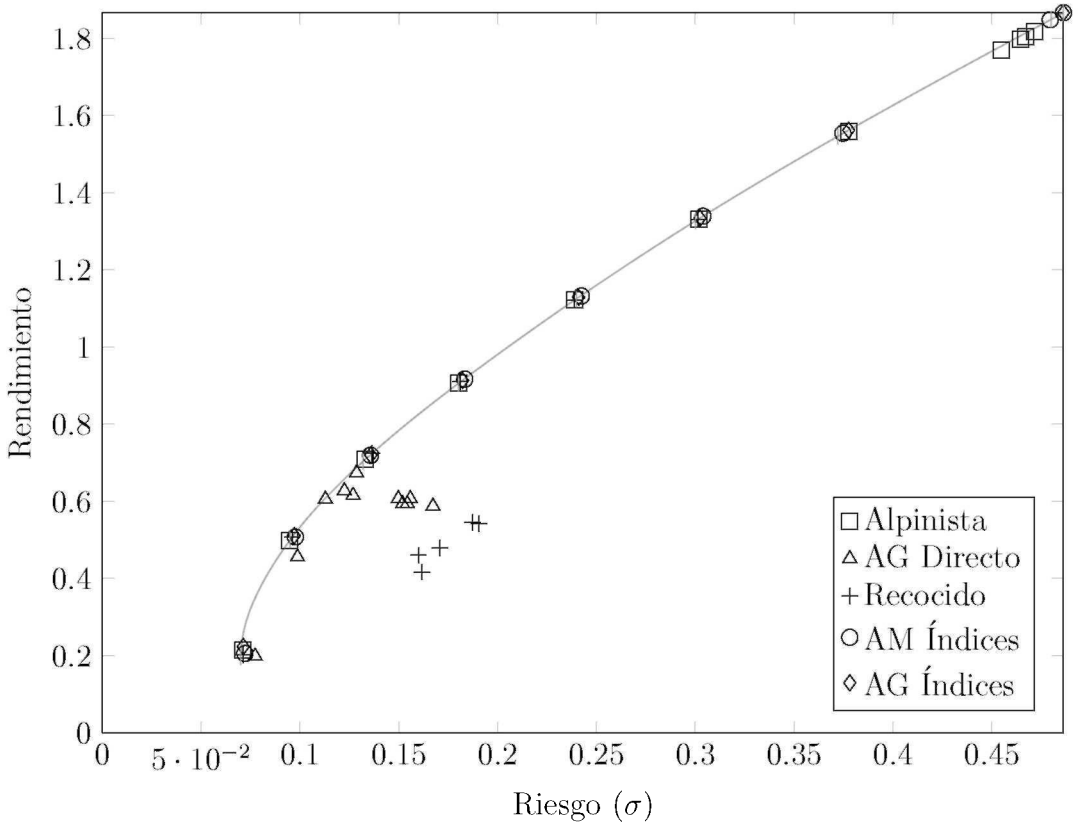

Figura 4.2: Puntos en el Diagrama RR encontrados por los algoritmos con cardinalidad restringida en un espacio de 100 acciones

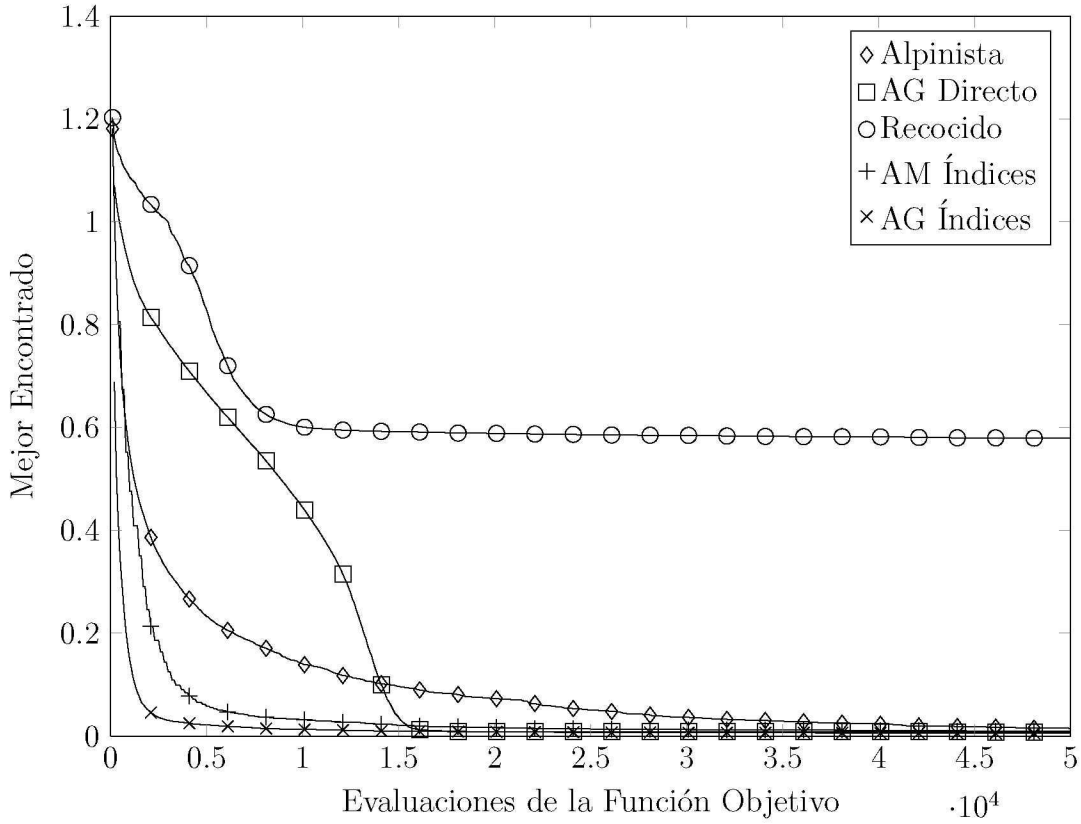

Figura 4.3: CME en el problema con 100 acciones sin restricción de cardinalidad

propósito de un algoritmo es llegar a cero, se puede ver que el único algoritmo que no llega al óptimo es el de recocido simulado. Así mismo, es interesante ver que al principio de la curva de este algoritmo, el crecimiento es más lento, y luego se acelera. Este punto de inflexión corresponde al valor en el que la temperatura es lo suficientemente baja para que el algoritmo deje de aceptar malas soluciones.

La evolución del AG de enfoque directo presenta un comportamiento similar, con la diferencia de que el retraso de la curva es mayor, pero el algoritmo sí logra encontrar el óptimo. En el caso del AG, este retraso puede ser causado por que la población no cuenta con los pedazos de solución suficientes para armar un portafolio óptimo, y el crecimiento se acelera cuando aparecen estos pedazos de solución en la población.

En la figura 4.4 se puede ver que mientras el comportamiento del algoritmo de búsqueda de alpinista mejora un poco al agregar la restricción de cardinalidad, el AG de enfoque directo no maneja nada bien esta restricción, pues pierde la capacidad de encontrar puntos cercanos al óptimo. El algoritmo de recocido simulado sigue teniendo un desempeño similar, aunque su evaluación mejora un poco.

Entre todos los algoritmos, el que tarda menos en encontrar una solución cercana al óptimo es el AG basado en índices. Se puede ver que ambos algoritmos basados en

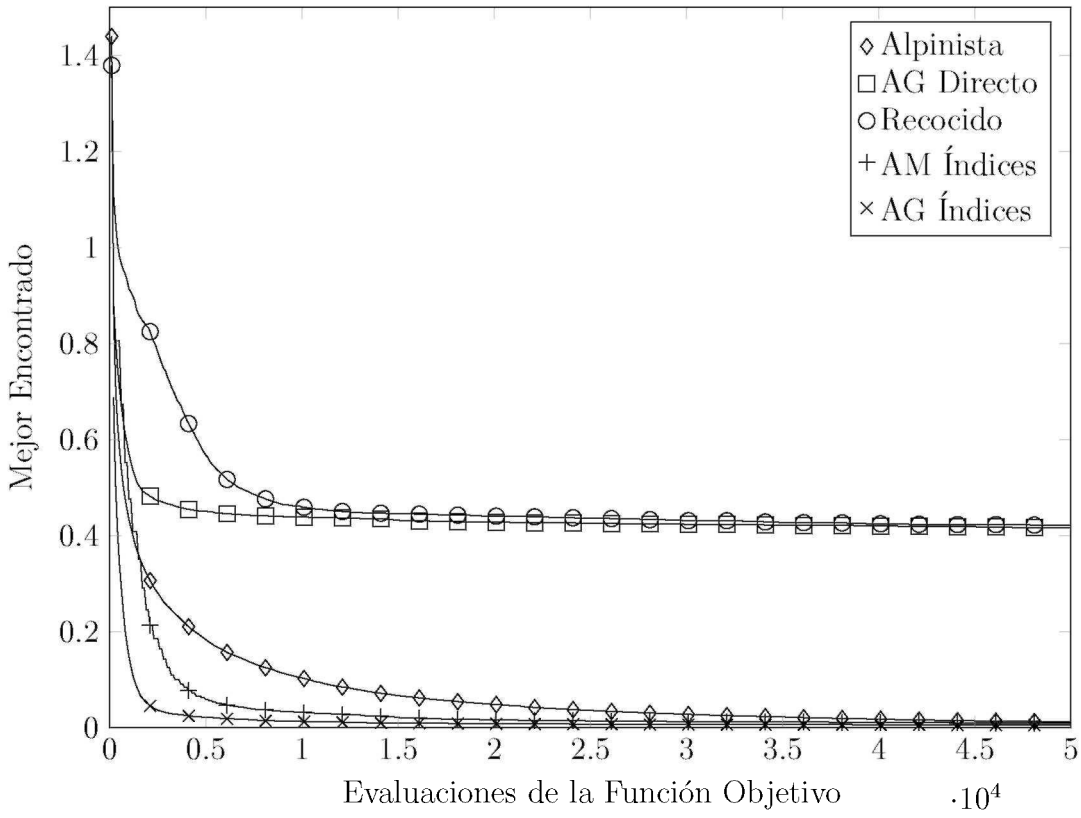

Figura 4.4: CME en el problema con 100 acciones con cardinalidad restringida

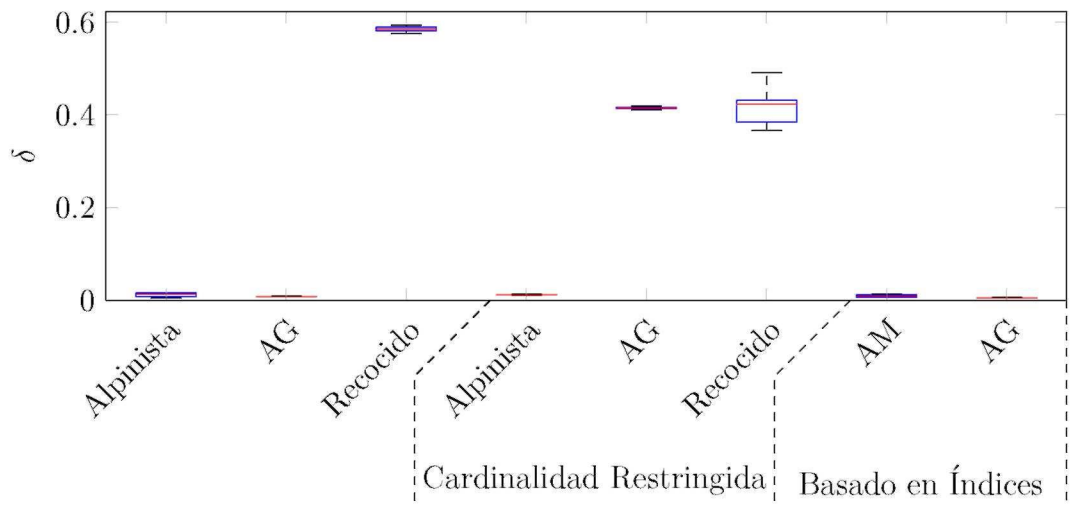

Figura 4.5: Calidad de las Soluciones encontradas por cada algoritmo, 100 acciones

índices se podrían haber parado a las 20×10^3 y habrían encontrado una solución casi tan buena como la solución final. En la figura 4.5 se puede ver como la mayoría de los algoritmos que llegan cerca del óptimo, lo hacen de manera consistente. Incluso los algoritmos con mal desempeño llegan a un valor similar en cada corrida, con excepción del algoritmo de recocido simulado con restricción de cardinalidad.

4.3. Resultados Obtenidos en un Espacio de 417 acciones

Se aumentó el número de acciones a 417. En la figura 4.6 se muestran mejores portafolios encontrados por los algoritmos con enfoque basado en índices y los algoritmos con enfoque directo sin restricciones. A diferencia de la figura 4.1 en que se muestra el diagrama equivalente pero con 100 acciones, en este caso el AG de enfoque directo no encuentra buenas soluciones para ningún valor de λ. El algoritmo de recocido simulado, que encontraba algunos puntos en la parte inferior de la curva, en este caso tampoco encuentra portafolios cercanos al óptimo. Incluso el algoritmo de búsqueda de alpinista deja de encontrar buenas soluciones después del cuarto valor de λ. Ambos algoritmos basados en índices encuentran puntos muy cercanos a los encontrados por programación cuadrática.

A pesar de que cuando se restringe la cardinalidad (figura 4.7) las soluciones encontradas por los algoritmos directos mejoran, el algoritmo de recocido simulado y el AG directo siguen encontrando puntos lejanos de la frontera de eficiencia salvo para el punto más bajo de la curva, en donde sólo se minimiza el riesgo. El algoritmo de

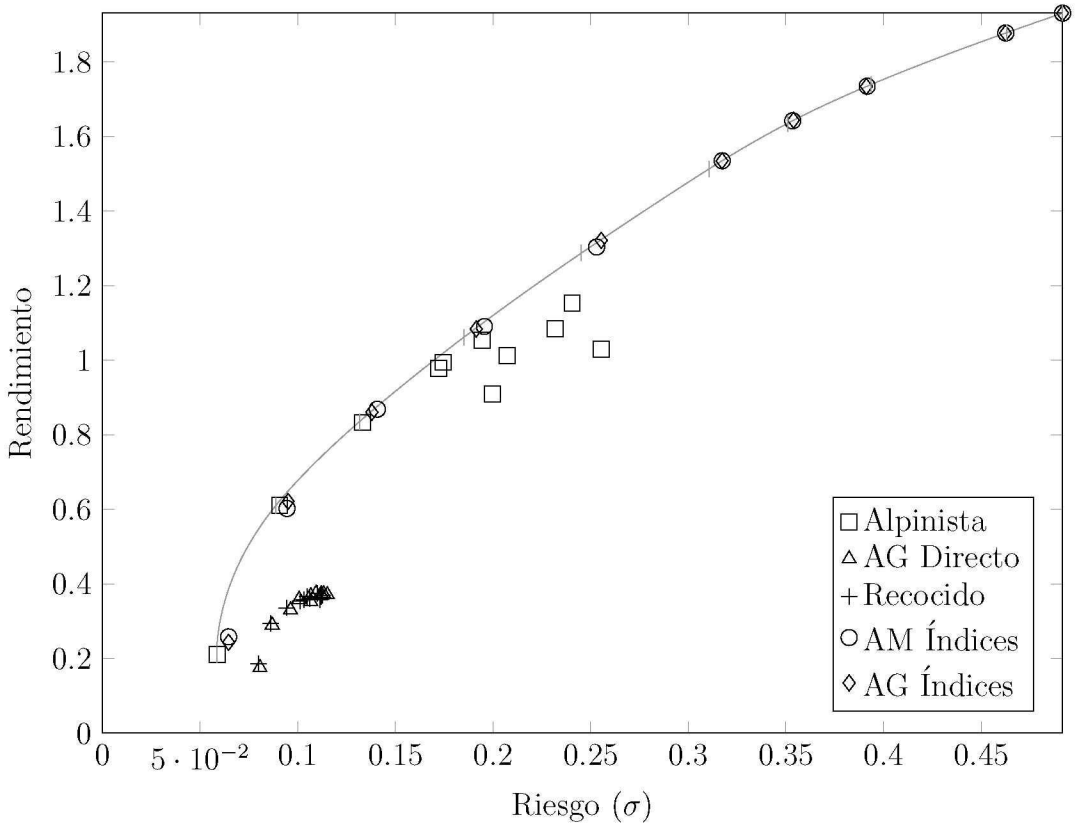

Figura 4.6: Puntos en el Diagrama RR encontrados por los algoritmos sin restricción de cardinalidad en un espacio de 417 acciones

búsqueda de alpinista se acerca más a la frontera de eficiencia, y obtiene puntos muy cercanos al óptimo para los primeros cinco valores de λ. Sin embargo no alcanza a llenar la frontera, y el resto de los puntos quedan cercanos entre sí.

Al observar la evolución en el tiempo de los algoritmos en las figuras 4.8 y 4.9 también se puede ver el mismo punto de inflexión en el algoritmo de recocido simulado que en su contraparte de 100 acciones, especialmente sin cardinalidad restringida. Aparentemente, el AG directo no logra obtener los pedazos de solución necesarios para encontrar portafolios cercanos al óptimo, y la evolución de las soluciones nunca se acelera. Se obtiene un resultado ligeramente mejor cuando la cardinalidad se restringe, que casi iguala al obtenido por recocido simulado. Sin embargo, tanto con restricción de cardinalidad como sin ella, el algoritmo no parece converger, lo que significa que podría obtener una mejor solución si se dejara correr un número mayor de evaluaciones de la función objetivo.

El algoritmo de búsqueda de alpinista mejora el desempeño de los otros dos algoritmos de enfoque directo, aunque de igual manera se queda en puntos subóptimos, como se pudo ver en los diagramas RR. Al igual que para 100 acciones, el algoritmo

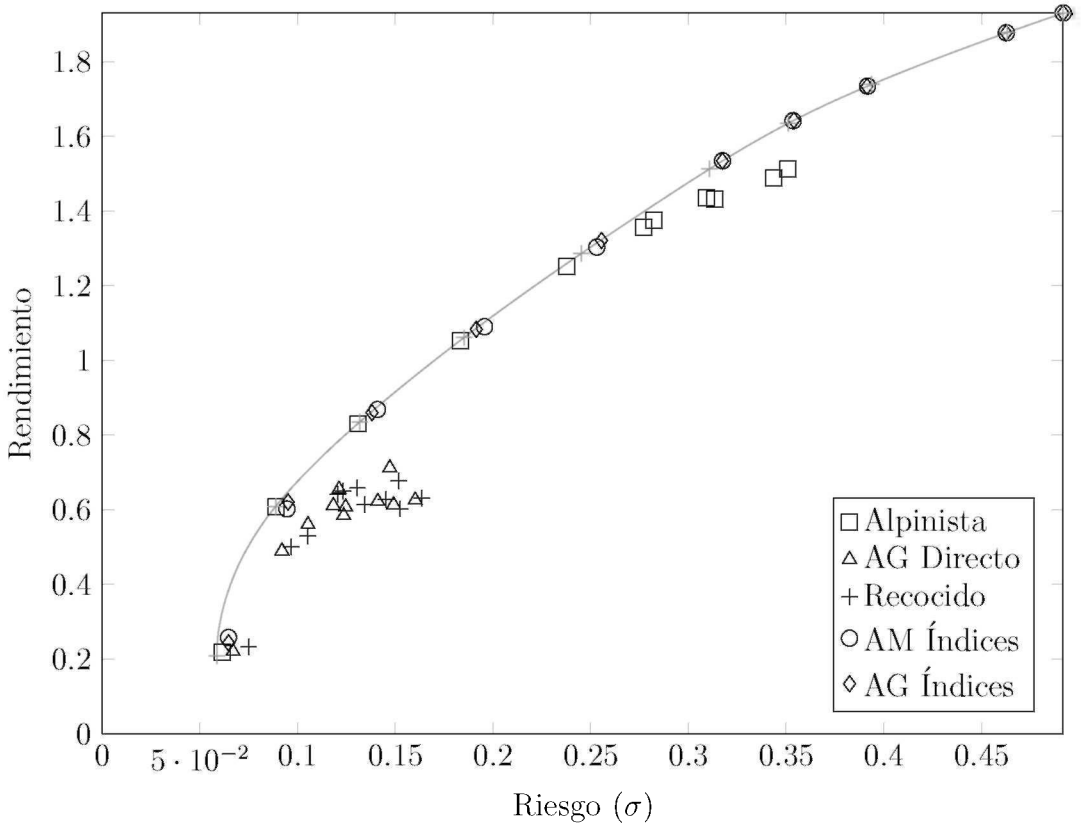

Figura 4.7: Puntos en el Diagrama RR encontrados por los algoritmos con cardinalidad restringida en un espacio de 417 acciones

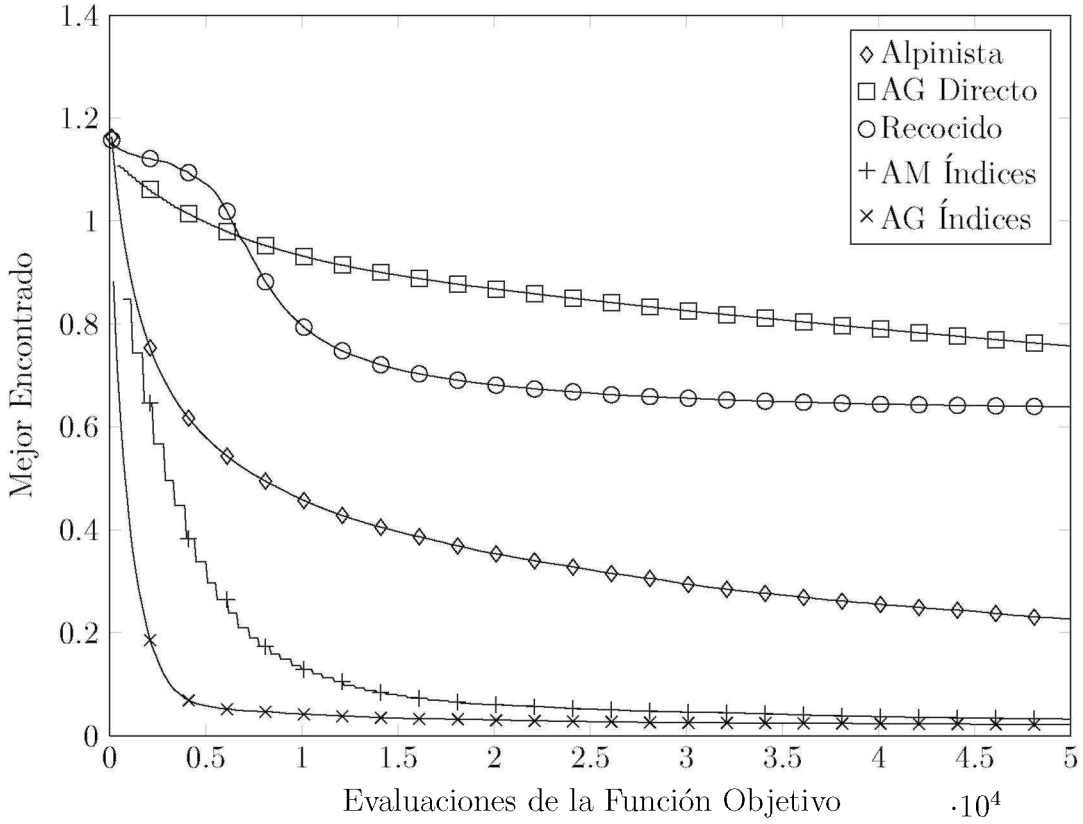

Figura 4.8: CME en el problema con 417 acciones sin restricción de cardinalidad

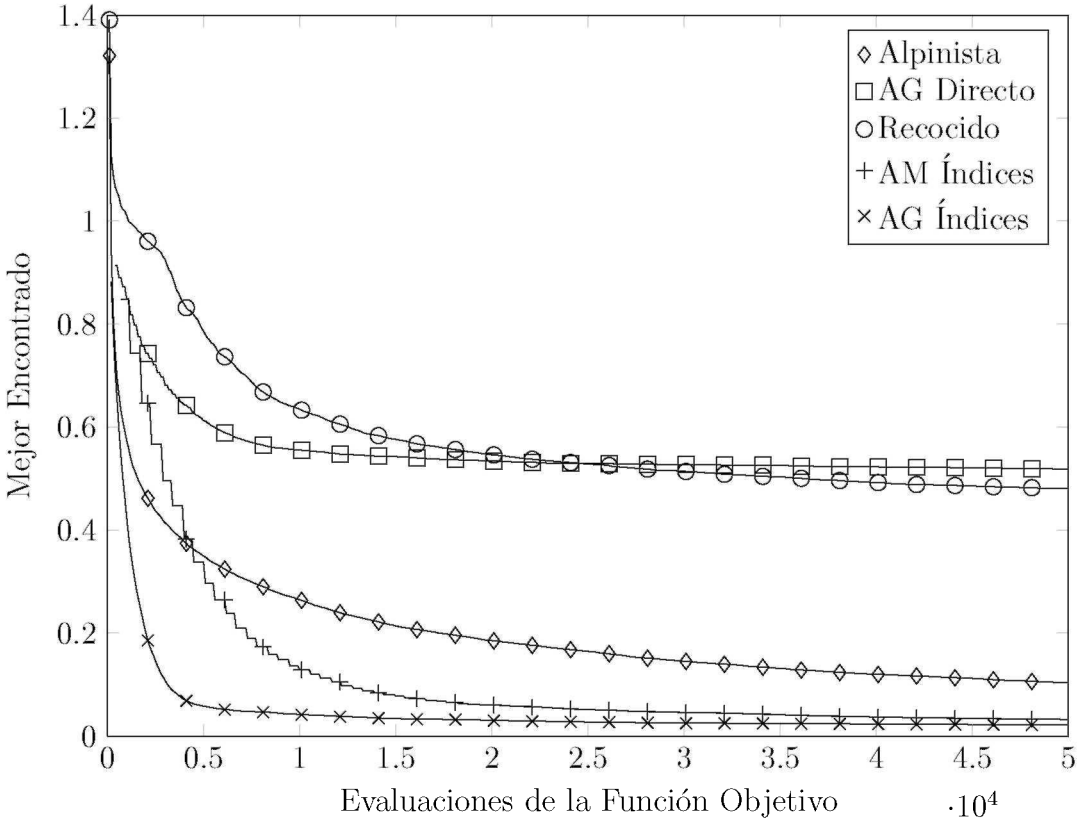

Figura 4.9: CME en el problema con 417 acciones con cardinalidad restringida

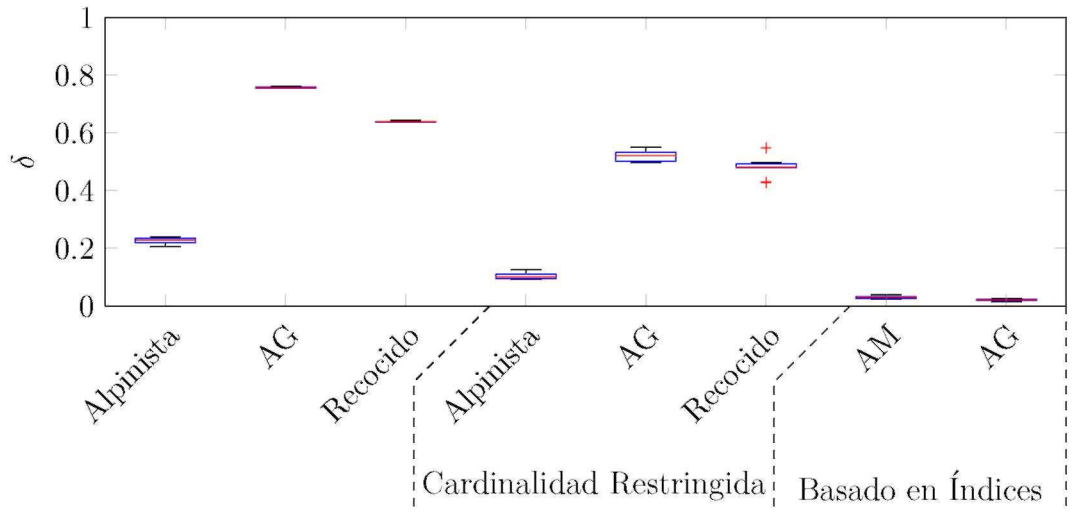

Figura 4.10: Calidad de las Soluciones encontradas por cada algoritmo, 417 acciones

que mejor desempeño tiene es el AG basádo en índices. Ambos algoritmos basados en índices tienen buen desempeño, y en ambos casos probablemente se podrían haber detenido antes de las 50×10^3. La figura 4.10 muestra como todas las soluciones obtenidas tienen poca variacion en las diferentes corridas.

4.4. Resultados Obtenidos en un Espacio de 1000 acciones

Debido a que el tiempo de ejecución por evaluación de la función objetivo del AG directo aumenta con el tamaño del espacio, y ya que la calidad de las soluciones no es competitiva con el algoritmos de búsqueda de alpinista y los basados en índices, este algoritmo se dejó de probar después de 417 acciones. Esta sección muestra los resultados obtenidos por el resto de los algoritmos en un problema con 1000 acciones.

Es importante notar en los diagramas RR (figuras 4.11 y 4.12), que la escala del eje de rendimiento cambió entre el problema con 417 acciones y el problema con 1000 acciones. Esto pasó por que la acción de máximo rendimiento en el problema con 1000 acciones tiene un rendimiento mayor que la acción de máximo rendimiento del problema con 417 acciones. Además, la escala del eje de riesgo es similar al eje de riesgo cuando sólo se toman 417 acciones. Esto significa que la mayor parte de la frontera de eficiencia con 1000 acciones contiene portafolios que dominan a los portafolios con 417 acciones.

La figura 4.11 muestra los resultados obtenidos por los algoritmos cuando la cardinalidad no se restringe. En este caso, se ve como casi todos los valores obtenidos por recocido simulado caen en un punto lejos de la frontera de eficiencia, salvo por tres va-

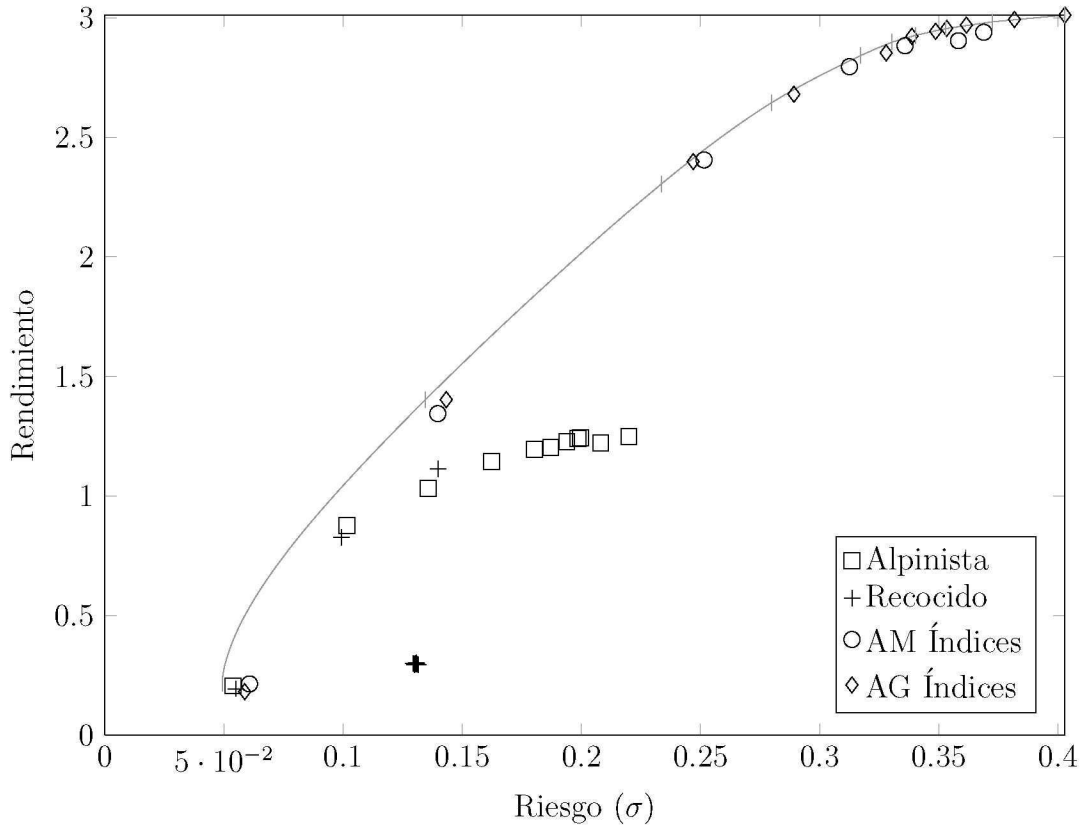

Figura 4.11: Puntos en el Diagrama RR encontrados por los algoritmos sin restricción de cardinalidad en un espacio de 1000 acciones

lores que se acercan. Los puntos encontrados por el algoritmo de búsqueda de alpinista se acercan un poco más, sin embargo no alcanzan a llegar a la frontera. Por otro lado, ambos algoritmos basados en índices distribuyen sus puntos muy cerca de la frontera de eficiencia. De estos dos últimos algoritmos, el AG basado en índices cubre más cercanamente todos los puntos de la curva encontrados por programación cuadrática, mientras el AM no alcanza a llegar al final de la curva al punto de mayor rendimiento.

El desempeño de ambos algoritmos con enfoque directo mejora cuando se incluye la restricción de cardinalidad. Especialmente el del algoritmo de búsqueda de alpinista, que cubre una parte más amplia de la curva. El algoritmo de recocido simulado también mejora, y sus soluciones ya no se encuentran concentradas en un sólo punto, pero no tanto como el algoritmo de alpinista.

Cuando se observa la evolución del algoritmo de búsqueda de alpinista en las figuras 4.13 y 4.14, se puede ver que en ambos casos el algoritmo no parece haber convergido a un valor, sino que sigue encontrando valores nuevos. Esto es particularmente cierto cuando se incluye la restricción de cardinalidad. Probablemente si se dejara correr algunas generaciones más, este algoritmo encontraría soluciones cercanas al óptimo.

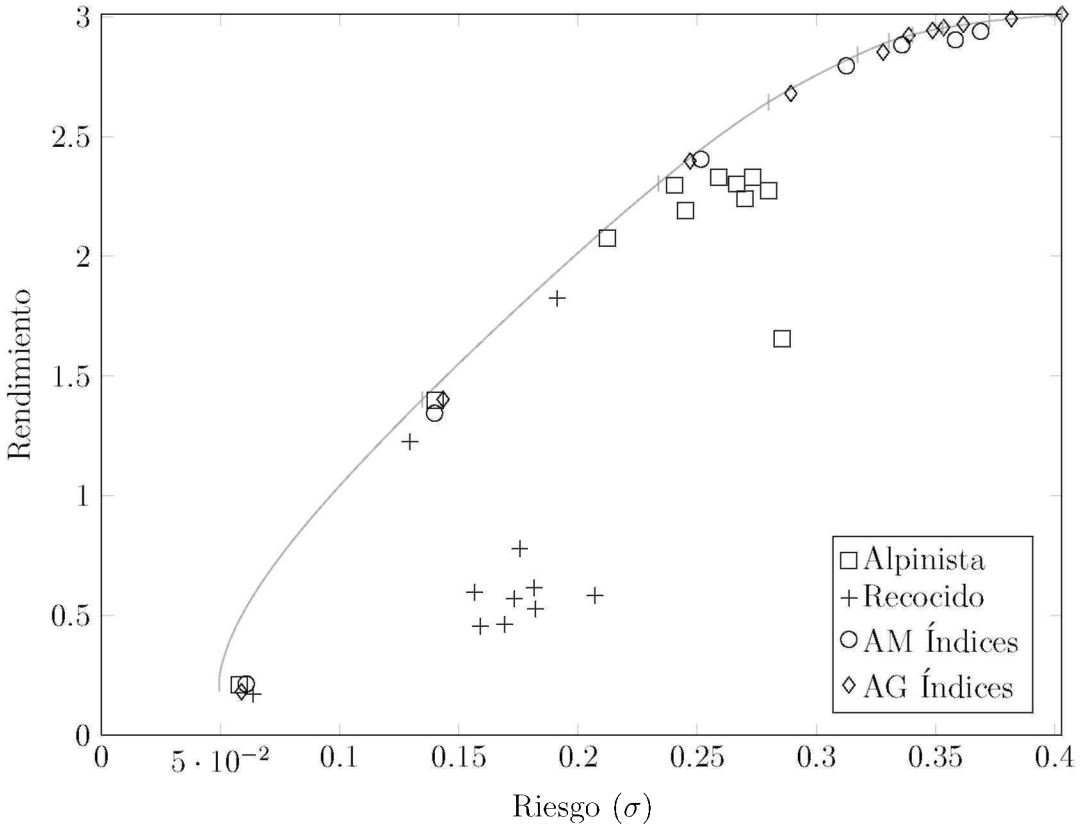

Figura 4.12: Puntos en el Diagrama RR encontrados por los algoritmos con cardinalidad restringida en un espacio de 1000 acciones

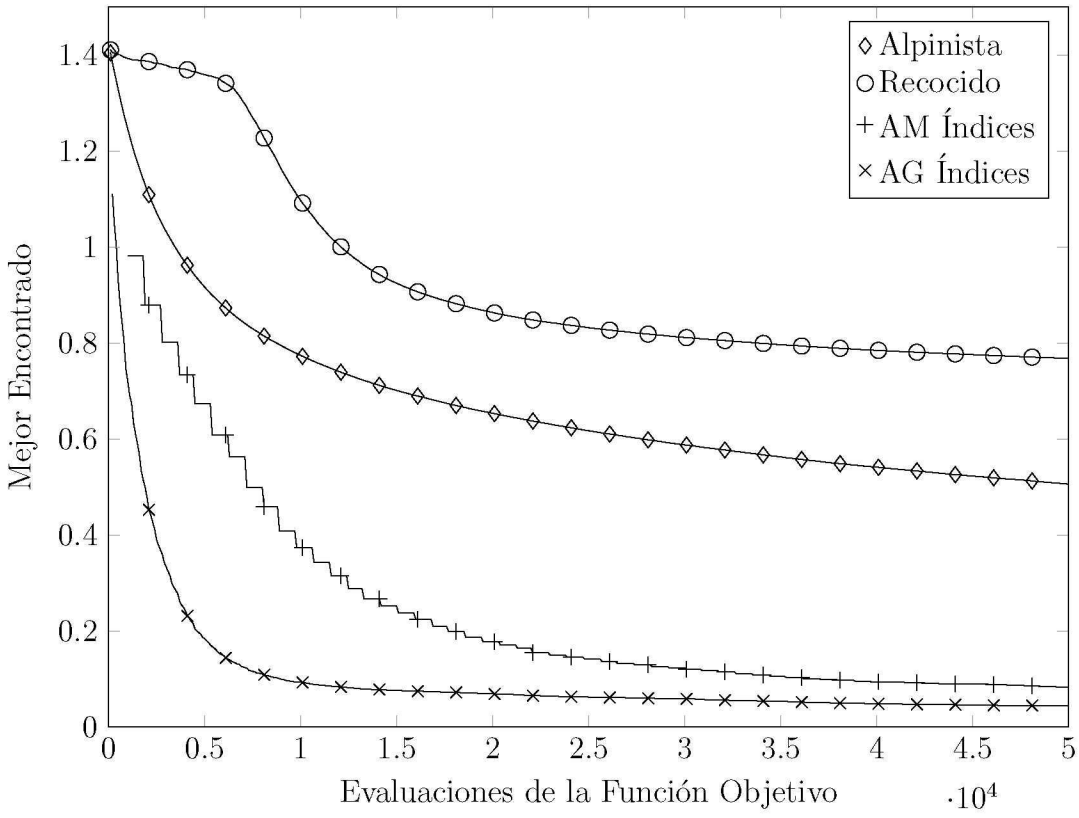

Figura 4.13: CME en el problema con 1000 acciones sin restricción de cardinalidad

Sin embargo, las evaluaciones permitidas fueron suficientes para que ambos algoritmos basados en índices consiguieran soluciones aceptables.

Se puede ver en la figura 4.15 como el AG basado en índices sigue siendo el mejor algoritmo, y como todos los algoritmos tienen poca variación en sus resultados. Es importante recordar que ambos algoritmos basados en índices buscan portafolios de cardinalidad restringida. Esta puede ser parte de la razón de que las soluciones obtenidas se alejen un poco del cero, y de los puntos encontrados por programación cuadrática. Generalmente se puede pensar que entre mayor sea el número de acciones en el espacio, mayor será la diferencia entre el óptimo sin restricción de cardinalidad, y el óptimo con cardinalidad restringida.

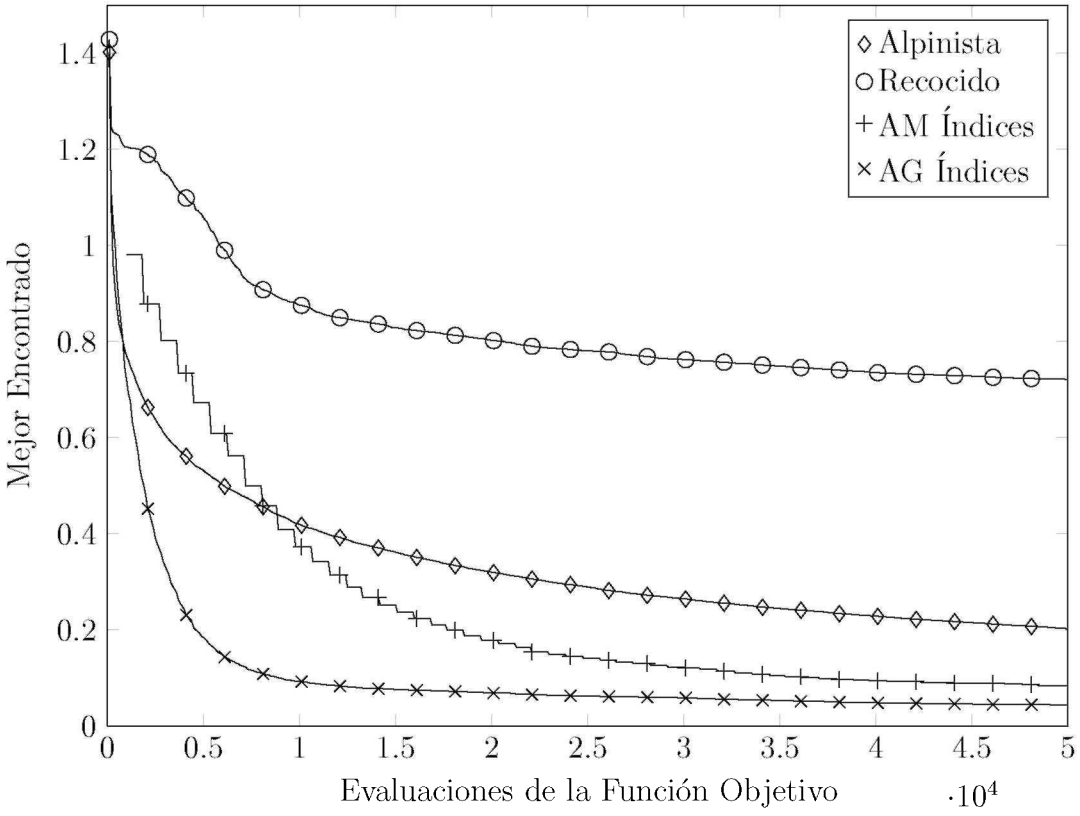

Figura 4.14: CME en el problema con 1000 acciones con cardinalidad restringida

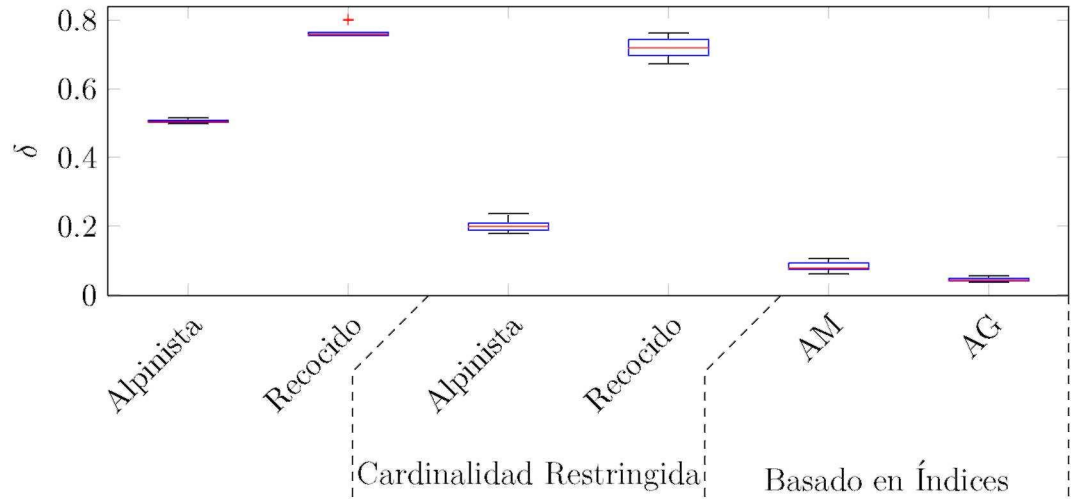

Figura 4.15: Calidad de las Soluciones encontradas por cada algoritmo, 1000 acciones

Capítulo 5

Conclusiones

En este trabajo se observó el desempeño de algoritmos evolutivos para encontrar portafolios óptimos en espacios grandes de acciones. Para poder resolver este problema, se utilizó una representación de un portafolio basada en índices Estos algoritmos se compararon con algoritmos de búsqueda local, y algoritmos evolutivos utilizando una representación directa.

5.1. Algoritmos con Enfoque Directo

Entre los algoritmos utilizados con enfoque directo, el que mejor desempeño obtuvo fue el algoritmo de búsqueda de alpinista. Este es un algoritmo sencillo, que no tiene forma de escapar de los óptimos locales. Es interesante además notar que el algoritmo de recocido simulado tiene un desempeño más pobre que el de búsqueda de alpinista en todos los casos. Se puede explicar este comportamiento si se recuerda que el algoritmo de recocido simulado es una generalización del algoritmo de búsqueda de alpinista, que cuenta con un mecanismo que le permite escapar de óptimos locales aceptando puntos subóptimos en el inicio del proceso de búsqueda. Sin embargo, en un problema sin óptimos locales, todo lo que este mecanismo hace es que el algoritmo brinque entre soluciones subóptimas sin avanzar hasta que la temperatura empieza a bajar tanto que en algoritmo se empieza a comportar como un algoritmo de búsqueda de alpinista.

Es importante aclarar que el hecho de que el algoritmo de recocido simulado tenga este desempeño, va estrictamente ligado a la función de vecindad que se utiliza. Lo que se concluye no es que el algoritmo de recocido simulado sea peor que el algoritmo de búsqueda de alpinista, sino que la función de vecindad utilizada favorece al algoritmo de búsqueda de alpinista como método para resolver este problema en particular.

El algoritmo genético con enfoque directo parece tener un buen comportamiento en espacios de pocas acciones sin restricción de cardinalidad. La curva de mejor encontrado sugiere que en este caso, el algoritmo comienza con un comportamiento pobre, hasta que llega un punto en el cual aparecen en la población las piezas de la población que se necesitan para poder armar buenas soluciones. Es en este punto que el algoritmo

acelera su crecimiento, e incluso le gana al algoritmo de búsqueda de alpinista en llegar al óptimo. Sin embargo, cuando el espacio de acciones crece, se vuelve cada vez más difícil que estos pedazos de solución aparezcan en la población.

5.2. Algoritmos con Representación Basada en Índices

Se propusieron dos algoritmos evolutivos con representación basada en índices, uno es un algoritmo genético y el otro es un algoritmo memético. Para todos los tamaños de espacio, se puede ver que este enfoque obtiene resultados más rápido que los algoritmos con enfoque directo. Entre estos dos, es el algoritmo genético el que llega a soluciones óptimas con menos evaluaciones de la función objetivo.

El algoritmo memético es una generalización del algoritmo genético que incluye un mecanismo de aprendizaje implementado como una heurística local. La heurística local utilizada fue un algoritmo de búsqueda de alpinista, que utiliza 10 evaluaciones de la función objetivo para tratar de encontrar mejores pesos que los que tiene el individuo original para ese conjunto de acciones. Los resultados obtenidos sugieren que este mecanismo de aprendizaje no mejora considerablemente la evaluación de los individuos, y estas 10 evaluaciones de la función objetivo por cada individuo son un gasto innecesario y que es mejor dedicarle esas evaluaciones al mecanismo evolutivo del algoritmo.

5.3. Orden de Crecimiento en el Tiempo de Ejecución

El orden de crecimiento de los algoritmos en términos de evaluaciones de la función objetivo es difícil de definir, pues depende del criterio de terminación que se esté usando. En este caso se dejaron correr los algoritmos por un número de evaluaciones predeterminado para poder realizar una mejor comparación de la calidad del portafolio resultante. Sin embargo, se puede ver en las curvas de mejor encontrado que conforme el número de acciones aumenta, se vuelve más claro como los algoritmos basados en índices tienen un mejor desempeño que los algoritmos con enfoque directo.

Una evaluación de la función objetivo de un algoritmo basado en índices tiene un orden de crecimiento mayor que la de un algoritmo con enfoque directo. En la evaluación de la función objetivo de un algoritmo basado en índices se extrae una submatriz de riesgos y un subvector de rendimientos del tamaño del portafolio k. La obtención del riesgo de este portafolio es la operación con mayor orden de crecimiento y es $O(k^2)$.

Para evaluar la función objetivo del algoritmo con enfoque directo sin cardinalidad

restringida se debe realizar la misma operación pero sobre una matriz de N^2 elementos, donde N es el tamaño del espacio. El orden de crecimiento entonces es de $O(N^2)$.

Cuando se restringe la cardinalidad disminuye el tiempo de ejecución, pues al igual que en la representación basada en índices, se puede evitar evaluar aquellas acciones que no se incluirán en el portafolio, y encontrar el riesgo en $O(k^2)$. Sin embargo, se deben extraer los k mayores pesos del portafolio antes de evaluarlo, lo que tiene un orden de crecimiento $O(N\log(N))$ o $O(Nk)$ dependiendo de si se ordena todo el portafolio, o si se busca el peso mayor k veces [3].

A menos que el tamaño del portafolio esperado sea similar al número total de acciones, generalmente se cumple que $O(N\log(N)) > O(k^2)$, lo que significa que evaluar la función objetivo para el enfoque directo toma más tiempo que con una representación basada en índices. Esto quiere decir que incluso si el comportamiento de un algoritmo basado en índices y un algoritmo con enfoque directo es equivalente en el sentido de que le toma la misma cantidad de evaluaciones de la función objetivo encontrar una buena solución, aún así debe tomar más tiempo al algoritmo con enfoque directo realizar estas evaluaciones.

5.4. Trabajos Futuros

Existen varios puntos que se pueden proponer como trabajos a futuro. Se habló en el capítulo 2 que un criterio que puede ser útil para optimizar portafolios en mercados con ciertas características es la relación de Sharpe. Se puede adaptar programación cuadrática para que encuentre portafolios óptimos de acuerdo a este criterio. Sin embargo, al igual que el enfoque usado en este trabajo, esto implica utilizar espacios pequeños de acciones, y no incluir la restricción de cardinalidad. Se podría probar el desempeño del algoritmo propuesto en este trabajo al encontrar portafolios que optimicen la relación de Sharpe.

Existen ademas diversos estimadores de riesgo más complejos que la varianza del portafolio, pero que no se utilizan por que no es posible formular un problema de programación cuadrática que los incluya. Tal es el caso de incluir el sesgo y la curtosis de la serie de tiempo de rendimientos del portafolio en la estimación del riesgo.

Así mismo, el contar con un método que permita optimizar un portafolio en un conjunto de acciones grande, permite evaluar la mejora en la frontera de eficiencia al utilizar todo el espacio de acciones como posibilidad, en vez de unas cuantas acciones seleccionadas según el rendimiento u otros factores más dudosos.

Bibliografía

[1] Claus C. Aranha y Iba Hitoshi. A tree-based GA representation for the portfolio optimization problem. *Genetic and Evolutionary Computation Conference, (GECCO)*, páginas 873 – 880, 2008.

[2] T.-J. Chang, N. Meade, J.E. Beasley, y Y.M. Sharaiha. Heuristics for cardinality constrained portfolio optimization. *Computers and Operations Research*, 27:1271–1302, 2000.

[3] Thomas H. Cormen, Charles E. Leiserson, Ronald L.Rivest, y Clifford Stein. *Introduction to Algorithms*. The MIT Press, 2a. edición, 2001.

[4] Edwin J. Elton, Martin J. Gruber, Stephen J. Brown, y William N. Goetzmann. *Modern portfolio theory and investment analysis*. Wiley, 2002.

[5] Eugene Fama. Random walks in stock market prices. *Financial Analysts Journal*, 21(5):55–59, septiembre-octubre 1965.

[6] David E. Goldberg. *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley, 1989.

[7] S. Kirkpatrick, C.D. Gelatt, y M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, mayo 1983.

[8] Renato D.C. Monteiro y Ilan Adler. Interior path following primal-dual algorithms. part ii: Convex quadratic programming. *Mathematical Programming*, 44(1-3):43–66, mayo 1989.

[9] Pablo Moscato. On evolution, search, optimization algorithms and martial arts, towards memetic algorithms. Technical Report 826, Caltech Concurrent Computation Program, California Institute of Technology, Pasadena CA 91125, USA, 1989.

[10] Carlos Joan Martínez Romero. Optimización de portafolios con restricciones empleando algoritmos genéticos. Tesis de Maestría, Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Monterrey, diciembre 2008.

[11] Stuart Russell y Peter Norvig. *Artificial Intelligence, A Modern Approach.* Prentice Hall, Englewood Cliffs, New Jersey, 2a. edición, 2003.

[12] William F. Sharpe, Gordon J. Alexander, y Jeffrey W. Bailey. *Investments.* Prentice Hall, 6a. edición, Octubre 1996.

[13] Felix Streichert, Holger Ulmer, y Andreas Zell. Evolutionary algorithms and the cardinality constrained portfolio optimization problem. En *Operations Research Proceedings 2003, Selected Papers of the International Conference on Operations Research (OR 2003)*, páginas 3–5. Springer, septiembre 2003.

[14] Felix Streichert, Holger Ulmer, y Andreas Zell. Evaluating a hybrid encoding and three crossover operators on the constrained portfolio selection problem. En *Congress on Evolutionary Computation*, páginas 932–939, 2004.

[15] Manuel Valenzuela-Rendón. The virtual gene genetic algorithm. En *Genetic and Evolutionary Computation*, volumen 2724/2003, página 215. Springer Berlin / Heidelberg, 2003.

[16] Darrel Whitley, V. Scott Gordon, y Keith Mathias. Lamarckian evolution, the baldwin effect and function optimization. En *Parallel Problem Solving from Nature (PPSN)*, volumen 866/1994 de *Lecture Notes in Computer Science*, páginas 5 – 15. Springer, 1994.