



OPTIMIZACIÓN ROBUSTA DE CARTERAS DE INVERSIÓN MEDIANTE SPEA2

Proyecto Fin de Carrera de
Ingeniería Informática

Septiembre de 2010

Alumna: Carmen Lozano Masdemont

Tutores: David Quintana Montero, Pedro Isasi Viñuela

AGRADECIMIENTOS:

No podría finalizar el proyecto sin acordarme de todas aquellas personas que han hecho posible su realización y que me han apoyado, ayudado y compartido conmigo tantas experiencias durante el largo período universitario.

Es justo en este preciso instante, al contemplar de cerca la tan ansiada meta, cuando se hace memoria de lo que se deja atrás, sabiendo que, a partir de ahora, todo va a ser muy diferente. Y pese a las ganas de terminar, queda siempre una sensación de añoranza y buen recuerdo, especialmente debido gracias a la gente que ha hecho que mi etapa de formación sea una de las más importantes de mi vida.

Así, en primer lugar, quiero agradecer este proyecto a mis tutores, especialmente a David, quien pacientemente ha sabido responder a mis cuantiosas dudas y tranquilizarme cuando creía que no podría finalizar a tiempo. También por hacerme descubrir un campo nuevo para mí y con el que he aprendido mucho.

Un lugar destacado en este apartado queda para mis padres, por su incondicional apoyo durante todos estos años, y esa gran paciencia demostrada para aguantar mis agobios y cambios de humor cuando las cosas no me salen como espero. Calificados como “santos” por la mayoría de mis amigos, no puedo dejar de agradecerles haberme brindado la oportunidad que ellos no tuvieron de estudiar una carrera, sacrificándose de tantas cosas para que mi hermana y yo pudiéramos completar nuestros estudios. Y junto a ellos tengo que nombrar también a mi hermana, la otra gran víctima de mis ya conocidos nervios. Pequeña en edad, pero mayor en todos los demás aspectos, siempre ha sabido proporcionarme la cordura y tranquilidad que me faltaba para saber afrontar que, a veces, uno no consigue las cosas a la primera.

Y el otro santo a quien le debo tantos agradecimientos es a Miki. Por todas las tardes que para poder quedar conmigo se venía a la biblioteca, o yo a su casa y me miraba trabajar. Por tantas dudas que me ha sabido responder, y su enorme paciencia conmigo. Porque si le hiciera caso en algunos de sus consejos seguramente me iría mejor. Porque sin él la realización de este proyecto hubiera sido menos interesante. Por dedicarme todo su tiempo y saber esperarme. Prometo compensarte por todo este tiempo.

Y, por último, no podría olvidarme de tantos amigos y compañeros que han hecho más amena mi estancia en la universidad. Quisiera dar las gracias a Christian, por sus siempre buenos consejos y optimismo; a Sergio, por estar dispuesto a echarme una mano siempre que lo necesite y preocuparse tanto por mí; y muy especialmente a Alberto, por preguntarme absolutamente todos los días por los nuevos avances, apoyarme, ayudarme en todo lo que estuviera a su alcance y “asesorarme” como yo le decía siempre (por ejemplo: si el color de las gráficas quedaban mejor sobre fondo negro o azul, una duda que me llevó un buen rato despejar). Me gustaría mencionar, con un enorme cariño, a Azahara, mi amiga del alma desde los 4 años, con quien tantos buenos y malos momentos he vivido y a quien hace 2 meses que no veo y eso que vive a 3 minutos de mi casa. Gracias por cogerme el teléfono y aguantar mis largas conversaciones de quejas, y saber apoyarme y animarme siempre. Gracias por confiar en mí más que yo. Y a Javi, el chico más listo que conozco y quien más me contagió la pasión por la Informática. Y también a José, a quien quisiera agradecerle estar siempre pendiente de que cumpliera con el horario de la biblioteca.

Y, por supuesto, mencionar a Rúa, Garci, Charlie, Mari Tere, Pedrito, Alfredo, Raúl, Javi García y Carlos Hita, por hacerme pasar tan buenos momentos en la Universidad y de fiesta, y compartir uno de los mejores viajes que seguramente haga nunca, y que probablemente en Febrero volvamos a repetir (esta vez a otro destino).

A todos ellos, mi más sincero agradecimiento.

RESUMEN

En el presente proyecto se muestra la aplicación de un algoritmo genético multiobjetivo al problema de la optimización de carteras de inversión. En el documento, se hace una introducción tanto a la naturaleza de los problemas de optimización, como a la Teoría de Carteras de inversión y los algoritmos genéticos multiobjetivo. Todo esto sirve de base para la resolución del siguiente problema específico: encontrar la cantidad idónea a invertir en cada una de las acciones de una cartera, atendiendo a objetivos de rendimiento, riesgo y robustez. El análisis de este tema se apoya en el uso de un framework que permite observar paso a paso el proceso de optimización, utilizando un algoritmo genético multiobjetivo (SPEA2). El uso del software permitirá obtener tanto resultados gráficos como numéricos. Este trabajo concluirá con un análisis de los resultados obtenidos para validar las formas de modelar la robustez propuesta.

Palabras Claves: Algoritmos genéticos, Algoritmos Genéticos Multiobjetivo, cartera de acciones, optimización, Teoría Moderna de Portafolio, metaheurísticas, framework, SPEA2.

ABSTRACT

This work shows the application of the theory of genetic algorithms to an optimization problem of a portfolio of shares. In this document, we will introduce the nature of the problem as well as the Portfolio Theory and multiobjective genetic algorithms. All this is used as the basis to solve the next problem: finding the best quantity to invest in each stock of a share portfolio, according to yield, risk and robustness criteria. During the analysis we will make use of a framework that allows observing step by step the optimization process, using one multiobjective genetic algorithm (SPEA2). Using the software we will obtain graphics and numeric results. The project will end with the analysis of the results obtained in order to validate the ways to model the robustness proposed.

Keywords: Genetic Algorithms, Multi-Objective Evolutionary Algorithm (MOEA), portfolio optimization, optimization, Modern Portfolio Theory, metaheuristics, framework, SPEA2.

ÍNDICE DE CONTENIDOS

1.Introducción.....	12
1.1 Resumen del proyecto/Descripción del problema.....	12
1.2 Objetivos.....	13
1.3 Metodología.....	14
1.4 Medios empleados.....	16
1.5 Estructura del trabajo	16
2. Estudio del Problema	19
2.1 El contexto del problema.....	19
2.2 El estado de la cuestión.....	23
2.2.1 Elementos de Optimización.....	24
2.2.1.1 Problemas de Optimización	24
2.2.1.2 Problemas de Optimización Multiobjetivo	24
2.2.2 Técnicas Recientes Para la Optimización de Carteras	27
2.2.3 Algoritmos Evolutivos.....	28
2.2.3.1 Algoritmos Genéticos.....	31
2.2.4 Optimización Multiobjetivo Basada en Algoritmos Evolutivos.....	32
2.2.5. Aplicaciones de los Algoritmos Genéticos Multiobjetivo.....	35
2.2.6 Algoritmos Genéticos Multiobjetivo en la Optimización de Carteras de Inversión	35
2.2.7 Herramientas para la Implementación de Algoritmos Evolutivos Multiobjetivo.....	36
3.Gestión de Proyecto Software	38
3.1 Plan de Trabajo.....	38
3.1.1 Identificación de Tareas	38
3.1.2 Estimación de Tareas	39
3.2 Alcance del proyecto.....	42
3.2.1 Definición del proyecto.....	42
3.2.2 Estimación de tareas y recursos	42
3.2.2.1. Recursos Humanos	42
3.2.2.2. Gastos de Hardware	43
3.2.2.3. Gastos de Software.....	44
3.2.2.4. Otros Gastos	44
Desarrollo del proyecto	47
4.1 análisis y diseño de la solución.....	47
4.1.2 análisis	48
4.1.2.1. Capacidades y restricciones generales:	48
4.1.3 plataforma y herramientas de desarrollo.....	52

Funcionamiento de la aplicación.....	54
4.1.4 Diseño de la aplicación	54
4.1.4.1. Clase Solution:	56
4.1.4.2. Clase Operator	57
4.1.4.3. Clase PROBLEM.....	58
4.1.4.4. Clase ALGORIThm.....	59
4.1.4.5. Algoritmos NSGA-ii y SPEA2	59
4.1.4.6. Clase Carterainversion.....	60
4.1.4.7. Clase Lecturadatosentrada	60
4.1.4.8. Interfaz experiment	61
4.1.4.9. Diseño de los operadores genéticos.....	63
4.1.4.10. Funciones objetivo:.....	72
4.1.4.11. Remuestreo de parámetros:	77
4.1.4.12. Fichero de entrada:.....	78
5. Experimentación.....	80
5.1. Muestra utilizada.....	80
5.2. Proceso de experimentación	81
5.2.1 Plan de experimentación	81
5.2.1.1 Experimento base	82
5.2.1.2 Sensibilidad al radio.....	82
5.2.1.3 Sensibilidad al exponente.....	82
5.2.1.4 Sensibilidad a remuestreo	83
5.2.1.5 Validación de las técnicas propuestas.....	83
5.2.1.6 Métricas	84
5.2.1.7 Análisis Estadístico de los resultados	87
5.2.1.8 Ajuste de parámetros y operadores.....	88
5.3 Análisis de los resultados.....	90
5.3.1. Selección del mejor algoritmo	91
5.3.2 Sensibilidad al radio	93
5.3.2.1. Resultados generales	93
5.3.2.2. Resultados agrupando por radio	95
5.3.2.3. Resultados agrupando por estabilidad.....	96
5.3.3. Sensibilidad al Exponente	98
5.3.2.4. Resultados Generales	98
5.3.2.5. Resultados agrupando por exponente	100
5.3.2.6. Resultados agrupando por estabilidad.....	101
5.3.4. Sensibilidad al remuestreo	102

5.3.4.1. Resultados generales	103
5.3.4.2. Análisis Agrupando por estabilidad	104
5.3.5. Validación de las técnicas propuestas	105
5.3.5.1. Resultados con diferentes exponentes y número de objetivos.....	106
5.3.5.2. Resultados con diferentes radios y números de objetivos	109
5.3.5.3. Resultados con remuestreo	112
6.Conclusiones	115
6.1 Objetivos alcanzados.....	115
6.2 Conclusiones finales:.....	115
6.3 Líneas Futuras de Investigación.....	117
6.4 Problemas encontrados.....	118
6.5 Opiniones personales.....	119
Anexo I: Seguimiento de proyecto de fin de carrera.....	122
Planificación inicial	122
Planificación final	124
Presupuesto inicial	126
Presupuesto final	126
Anexo II: Manual de usuario.....	128
¿Para qué sirve la aplicación?.....	128
Requisitos mínimos:	128
Formato del fichero de entrada.....	129
Párametros de configuración:	130
Ejecución de la aplicación	132
Datos de Salida:	136
Modo de ejecución con el script R:.....	138
Ejemplo de los ficheros de salida:	140
Bibliografía.....	141
Glosario.....	143

ÍNDICE DE FIGURAS

Figura 1: Metodología de trabajo	15
Figura 2: Frente de Parte de una función con dos objetivos	26
Figura 3: Estructura de un cromosoma.....	30
Figura 4: Representación de genotipo y fenotipo.....	30
Figura 5: Esquema de la evolución	30
Figura 6: Representación binaria de un individuo	31
Figura 7: Pseudocódigo de un algoritmo genético.....	32
Figura 8: Esquema del funcionamiento de la aplicación	54
Figura 9: Esquema del funcionamiento de la aplicación	55
Figura 10: Componentes utilizados para representar la solución	56
Figura 11: Código de la clase RealSolutionType, para representar variables reales	57
Figura 12: Esquema de un Algoritmo Evolutivo para Optimización Multiobjetivo	59
Figura 13: Pseudocódigo del algoritmo SPEA2.....	60
Figura 14: Pseudocódigo del algoritmo NSGA-II.....	60
Figura 15: Código para clasificar en función de la estabilidad	62
Figura 16: Código para la obtención del frente no dominado.....	62
Figura 17: Código para obtener los individuos pertenecientes al frente no dominado	63
Figura 18: Representación problema: diseño del cromosoma	64
Figura 19: Esquema del cruce.....	66
Figura 20: Código para rectificar individuos que no sumen uno	68
Figura 21: Código para rectificar individuos que no cumplan la restricción de cardinalidad.....	70
Figura 22: Código de la mutación	72
Figura 23: Código para la asignación de objetivos	73
Figura 24: Creando la matriz de covarianzas	74
Figura 25: Usando la librería Commons-Math para crear la matriz de covarianzas.....	74
Figura 26: Código para la evaluación del riesgo	75
Figura 27: Código para la evaluación del rendimiento	76
Figura 28: Ejemplo comparación de algoritmos	85
Figura 29: Ejemplo II de comparación de algoritmos	85
Figura 30: Ejemplo representación de la dispersión	86
Figura 31: Hipervolumen cubierto por las soluciones no dominadas	86
Figura 32: Análisis estadístico de los experimentos	88
Figura 33: Comparación de los algoritmos multiobjetivo.....	92
Figura 34: Comparación de los algoritmos multiobjetivo.....	92
Figura 35: Gráfica comparativa de los resultados en función del radio	95
Figura 36: Análisis estadístico agrupado por estabildades	96
Figura 37: Análisis gráfico	100
Figura 38: Análisis gráfico agrupados por estabilidad	101
Figura 39: Análisis gráfico cuarto experimento	104
Figura 40: análisis gráfico diferentes estabildades y número de objetivos	108
Figura 41: Análisis gráfico resultados experimento con distintos rangos de variación.....	111
Figura 42: Análisis gráfico resultados experimento con remuestreo	113
Figura 43: Planificación inicial	123
Figura 44: Planificación final.....	125
Figura 45: Ejemplo de fichero de datos válido	130
Figura 48: Menú de la aplicación	132
Figura 46: Ejemplo de parámetros de configuración	132
Figura 47: Ejecución de la aplicación	132
Figura 49: Ejemplo salida ejecutando la tercera opción	133

Figura 50: Ejemplos configuración mediante la opción 4.....	135
Figura 51: Ejemplo salida opción 5.....	136
Figura 52: Salida de la ejecución de la aplicación.....	136
Figura 53: Traza mostrada durante la ejecución.....	137
Figura 54: Estructura de la carpeta creada.....	137
Figura 55: Contenido de la carpeta DATA.....	138
Figura 56: Menú archivo de script R.....	139
Figura 57: Menú de la aplicación de script R.....	139
Figura 58: Ejemplo salida del fichero FUN.....	140
Figura 59: Ejemplo de salida del fichero VAR.....	140

ÍNDICE DE TABLAS

Tabla 1: Descripción estimada por tareas y fases.....	40
Tabla 2: Desglose del coste por hora según el rol del trabajador.....	43
Tabla 3: Desglose del coste en Hardware.....	43
Tabla 4: Desglose del coste en Software.....	44
Tabla 5: Desglose de los costes asociados con el desarrollo del proyecto.....	45
Tabla 6: Presupuesto total del proyecto.....	45
Tabla 7: Formato de las tablas resultantes de los contrastes estadísticos.....	88
Tabla 8: Parámetros de los algoritmos genéticos.....	90
Tabla 9: Resultados primer experimento.....	91
Tabla 10: Resultados de los contrastes de hipótesis.....	93
Tabla 11: Resultados segundo experimento.....	94
Tabla 12: Análisis estadístico caso base.....	95
Tabla 13: Análisis estadístico radio doble.....	95
Tabla 14: Análisis estadístico radio triple.....	96
Tabla 15: Análisis estadístico estabilidad alta.....	97
Tabla 16: Análisis estadístico estabilidad media.....	97
Tabla 17: Análisis estadístico estabilidad baja.....	97
Tabla 18: Resultados tercer experimento.....	99
Tabla 19: Análisis estadístico caso base.....	100
Tabla 20: Análisis estadístico exponente al cuadrado.....	100
Tabla 21: Análisis estadístico exponente al cubo.....	101
Tabla 22: Análisis estadístico estabilidad alta.....	101
Tabla 23: análisis estadístico estabilidad media.....	101
Tabla 24: Análisis estadístico estabilidad baja.....	102
Tabla 25: Resultados cuarto experimento.....	104
Tabla 26: Estadísticos Remuestreo frente a caso base.....	105
Tabla 27: Resultados experimento diferente número de objetivos.....	106
Tabla 28: análisis de la robustez con el exponente.....	107
Tabla 29: análisis de la robustez con el radio.....	109
Tabla 30: Comparativa análisis de la robustez con el radio.....	110
Tabla 31: Resultados con remuestreo.....	112
Tabla 32: Análisis estadístico remuestreo.....	112
Tabla 33: Presupuesto inicial.....	126
Tabla 34: Gastos finales.....	126
Tabla 35: Parámetros de configuración.....	132

Capítulo 1

Introducción

1. INTRODUCCIÓN

Este Proyecto Fin de Carrera de la titulación de Ingeniería Informática tiene como finalidad diseñar, implementar y experimentar con algoritmos que proporcionen las mejores combinaciones de acciones para conseguir simultáneamente buenas rentabilidades, con un nivel aceptable de riesgo y con cierta predictibilidad en los rendimientos (estabilidad/robustez). Este capítulo introductorio tiene como propósito poner al lector en situación, ofreciendo una visión global acerca del proyecto desarrollado, y descrito en los sucesivos capítulos, así como una declaración de los objetivos que se persiguen con su realización.

Además, esta sección recoge la metodología de trabajo seguida y cómo se estructura el presente documento.

1.1 RESUMEN DEL PROYECTO/DESCRIPCIÓN DEL PROBLEMA

Las carteras de inversión se componen de un conjunto de activos financieros o valores (bienes tangibles e intangibles), cada uno de los cuales tiene un riesgo y una rentabilidad. Estos elementos, considerados en conjunto, ofrecen la posibilidad de obtener mejores relaciones rentabilidad/riesgo a las logradas invirtiendo en instrumentos financieros de forma individual [1].

Aunque a primera vista pueda parecer sencillo, seleccionar una cartera de inversiones que se ajuste al perfil de cada inversor y a la situación del mercado en un momento determinado, no es una tarea fácil. Por este motivo, existen profesionales que se dedican exclusivamente a crear y gestionar estas carteras de productos financieros. Su manejo consiste en equilibrar las pérdidas y ganancias, repartiendo y compensando el riesgo de los diferentes instrumentos financieros entre los mismos. [2]

Un problema de optimización de cartera implica la asignación de inversiones a un número de diferentes bienes (activos) para maximizar el rendimiento y minimizar el riesgo en un período de inversión dado.

La optimización de problemas multiobjetivo intenta resolver problemas con más de un objetivo (en este estudio, estos objetivos son la maximización del beneficio, y minimización de la inestabilidad y riesgo de los activos que componen la cartera), posiblemente conflictivos entre ellos. En este tipo de problemas, a menudo no existe una solución única capaz de optimizar simultáneamente todos los objetivos, sino que la búsqueda tiende a producir un conjunto de alternativas que resulten deseables en un sentido u otro, conocidas como el frente de Pareto [3].

En este proyecto se aplican algoritmos genéticos multiobjetivo para resolver la optimización de carteras, con algunas limitaciones realistas, como son las restricciones de cardinalidad, el número máximo de activos a invertir y los límites superiores e inferiores de inversión permitidos.

Un problema fundamental de la optimización de carteras es su sensibilidad a los errores de estimación sobre el rendimiento futuro de los activos. Los perfiles de riesgo-beneficio de

una cartera varían significativamente ante cualquier cambio producido en el mercado (por ejemplo cuando se incrementan a corto plazo los tipos de interés). Si no se consigue la robustez, las soluciones simplemente resultan poco fiables, lo que deja al algoritmo únicamente útil como herramienta de análisis de datos históricos, sin la posibilidad de servir como una herramienta de decisión orientada al futuro [4]. Por este motivo, se pretende construir un sistema tolerante a pequeñas equivocaciones, capaz de realizar predicciones lo más ajustadas posibles a la realidad, incluso ante errores de estimación de los parámetros del modelo.

1.2 OBJETIVOS

La finalidad de este proyecto es desarrollar un algoritmo que resuelva el modelo de optimización para encontrar soluciones al problema de carteras de inversión robustas. Se concentrará la atención en la selección de los porcentajes a invertir en cada acción, con el fin de encontrar la mejor combinación de acciones considerando el rendimiento esperado de toda la cartera y el riesgo de la misma, cuya variabilidad, a lo largo del tiempo, sea lo menor posible.

Se considerarán para su resolución entornos de mercado reales. Estas restricciones pueden ser consecuencia de ciertas limitaciones prácticas para realizar transacciones en un mercado real, o reflejar preferencias del inversor. En concreto, se considerarán para su inclusión restricciones de los siguientes tipos [5]:

- Restricciones de cardinalidad: limitan el número de productos distintos que pueden incluirse en la cartera de inversión, por razones de supervisión o diversificación, o razones de control de coste de transacción. Pueden expresarse de la siguiente manera:

$$C_l \leq \sum_{l=1}^n b^l \leq C_u,$$

donde $b_i = 1$ si $x_i > 0$ y $b_i = 0$, en cualquier otro caso; C_l y C_u son el menor y mayor número de activos, respectivamente, requeridos para incluir en una cartera.

restricciones que aparecen habitualmente en

La **definición matemática** de estos parámetros (rendimiento r_p y riesgo σ), se define de la siguiente manera:

$$\sigma_p^2 = \sum_i \sum_j x_i x_j \sigma_{ij}$$

$$\sigma_p = \sqrt{\sigma_p^2}$$

$$r_p = \sum_i x_i r_i$$

$$\sum_i x_i = 1$$

Donde σ_{ij} es la covarianza entre el activo i y j ; si $i = j$, es la varianza del activo i .

σ_p^2 es la varianza de la cartera de activos y σ_p la desviación estándar del rendimiento esperado.

r_i es el rendimiento esperado del activo i .

r_p es el rendimiento esperado de la cartera.

$$x_i \in \mathcal{R}_0^+ \quad \forall_i$$

Siendo $x_1, x_2, \dots, x_{n-1}, x_n$ los porcentajes del capital invertido en cada acción.

- Restricciones de los umbrales mínimos y máximos para la inversión en un activo determinado: definen los límites inferiores y superiores de la proporción de cada activo que se puede tener en una cartera. Estas restricciones pueden resultar de la política institucional a fin de diversificar la cartera y para descartar inversiones de una entidad tan pequeñas que los costes de gestión superen a los beneficios potenciales. Se puede expresar matemáticamente como sigue:

$$f_i \leq x_i \forall_i$$

En resumen, para alcanzar esta meta, se han fijado los siguientes objetivos:

- La determinación de los pesos, por parte de los algoritmos genéticos, que maximicen el rendimiento esperado, teniendo en consideración el riesgo implicado y la necesidad de obtener soluciones robustas ante errores de estimación de los parámetros del modelo.
- La aplicación de técnicas de algoritmos genéticos multiobjetivo que nos permitan conseguir el objetivo anterior. Dichas técnicas deberán tener en cuenta las restricciones ya explicadas.
- Un análisis de los resultados obtenidos que determinen qué algoritmo se ajusta mejor a las características del problema. Para ello, se comparará la calidad de las soluciones obtenidas.

1.3 METODOLOGÍA

Con el fin de desarrollar un buen Proyecto Fin de Carrera, es necesario establecer una metodología de trabajo y ajustarse a ella. La metodología seguida se divide en cuatro fases diferenciadas, las cuales se describen a continuación:

- **Planteamiento del problema:** en esta fase el tutor me explica en qué consiste el proyecto a implementar, sugiriéndome la tecnología a usar y la documentación a leer para entender mejor el dominio del problema. En esta fase se producen reuniones con el tutor, con el fin de centrar el problema y analizar las posibles soluciones, documentación, herramientas y lenguajes, que se plantean. De entre todas estas posibilidades, se tratará de elegir las más cómodas para trabajar, siempre y cuando permitan alcanzar los objetivos perseguidos.
- **Fase de documentación sobre el problema:** al abordar un proyecto de cuyo dominio no se tiene pleno conocimiento, como es el de las Finanzas (rama de Economía) en este caso, y más concretamente el de las acciones y carteras de inversión, fue necesario leer bastante documentación para comprender no sólo qué se quería implementar, sino también la finalidad del propio proyecto. Asimismo, fue necesario documentarse también acerca de los algoritmos multiobjetivo y herramientas a utilizar para la implementación de la solución. La conclusión de esta etapa permitía la construcción de un planteamiento general del problema a abordar y del enfoque a seguir.

- **Desarrollo de la solución:** en esta fase se implementará la aplicación. Una vez decididas las tecnologías que son más adecuadas para el desarrollo del proyecto, éstas se utilizarán para la implementación del mismo. Ésta es quizá, la fase más complicada del proyecto, ya que conlleva bastante tiempo el desarrollar toda la aplicación conforme a los objetivos que se pretenden alcanzar. Además, requiere de un proceso de adaptación y aprendizaje de las tecnologías, que suele ser difícil de estimar de antemano.
- **Experimentación y evaluación de la solución:** en esta fase se establecen y se ejecutan las pruebas sobre la aplicación para constatar, no solo que cumple con la funcionalidad requerida, sino también para tratar de averiguar qué algoritmo y bajo qué parámetros se comporta mejor el programa. De esta forma se puede asegurar la validez de la solución desarrollada.
- **Desarrollo de la documentación:** esta fase recoge el proceso de elaboración de la presente memoria, recopilando las cuestiones más importantes referentes al proyecto. Se desarrollará a la vez que el resto de fases identificadas, ya que no es necesario esperar a haber evaluado la aplicación, para poder escribir parte de la misma. De esta forma se redacta la memoria teniendo más frescos los conceptos leídos en la fase de documentación, así como los relativos al desarrollo de la aplicación, haciéndose más llevadero todo el proceso.

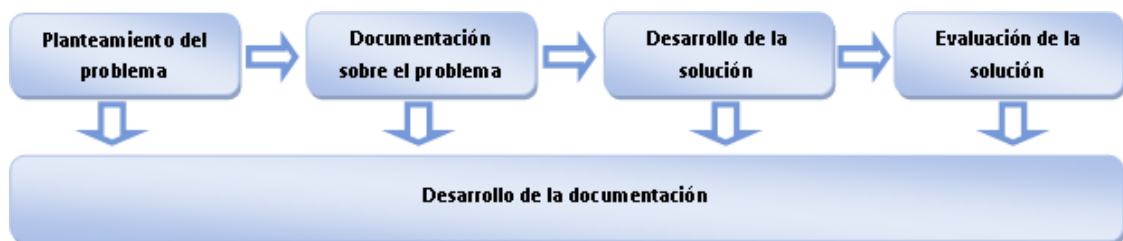


Figura 1 : Metodología de trabajo

El definir y seguir una metodología no es suficiente para llevar el proyecto a buen puerto, también es necesario establecer una planificación realista donde se sitúen las distintas fases a desarrollar. Esta planificación se define en el momento del planteamiento del problema, teniendo presente cuál va a ser la fecha aproximada de entrega del proyecto.

Aunque es muy difícil cumplir estrictamente los plazos marcados, ya que durante el desarrollo del proyecto surgen complicaciones imprevistas, mientras que otras tareas para las se calculaba más tiempo resultan más sencillas de lo esperado, se ha de intentar seguir de la manera más fiel posible. Al final de este documento se recogen tanto la planificación inicial como la real, para poder comparar la diferencia existente entre la planificación real y la estimada inicialmente y analizar qué tareas han llevado más tiempo del previsto y cuáles menos.

1.4 MEDIOS EMPLEADOS

Para la implementación del proyecto se ha hecho uso de un *framework* (*JMetal*), desarrollado bajo la tecnología Java (en su versión 6.0 del SDK). Al tratarse de un lenguaje muy utilizado, existen muchas librerías desarrolladas en él para muy diversos ámbitos. Para facilitar la codificación de ciertas funciones se ha importado una de estas librerías que proporciona *Apache Commons* (*Commons-Math*) en este lenguaje. El entorno de programación empleado para desarrollar el proyecto ha sido *Eclipse*. Las decisiones que nos han llevado a la elección de estas tecnologías y no otras, se argumentarán en el punto [4.1.3](#) (Plataforma y herramientas de desarrollo) de la fase de desarrollo.

Se han necesitado otras herramientas auxiliares para completar el desarrollo del proyecto, tanto para desarrollar la memoria (el paquete *MS Office*, *MS Project* para realizar la planificación, y la herramienta *Altova UModel 2008* para el diseño del diagrama de clases), como para el análisis de los resultados (un intérprete del lenguaje *R*, para facilitar la realización de contrastes estadísticos necesarios para la fase de experimentación)

La mayor parte de la información necesaria para la comprensión del problema ha sido facilitada por el tutor, así como los índices de valores utilizados como base para la implementación y posterior análisis de este proyecto.

1.5 ESTRUCTURA DEL TRABAJO

Este trabajo se ha estructurado en torno a seis capítulos principales, que se acompañan a su vez de una sección de bibliografía y de varios anexos que aparecen al final del presente documento. A continuación se indican los capítulos de los que se compone este proyecto y se ofrece una breve descripción de los mismos con el fin de orientar al lector durante su lectura:

- **Introducción:** en este capítulo se presenta el proyecto, explicando el problema que ha dado lugar al trabajo, los objetivos que pretende satisfacer y la metodología seguida para su desarrollo.
- **Estudio del problema:** este capítulo aborda el contexto en el que surge el problema, haciendo un estudio de las distintas tecnologías existentes para hacerle frente y finalmente planteando, de manera formal, el problema que se ha de resolver mediante el presente proyecto, así como las limitaciones a las que habrá que hacer frente.
- **Gestión del proyecto:** este capítulo se dedica al proceso de planteamiento, ejecución y control del proyecto para alcanzar un correcto equilibrio en cuanto a coste, plazos y calidad. Además se hace un análisis de los posibles riesgos que pudieran surgir durante el desarrollo del proyecto y cómo hacerles frente. Finalmente, se establece un plan de pruebas para asegurar que el sistema cuenta con la calidad requerida y que se satisfacen todos los objetivos pedidos.
- **Sistema desarrollado:** aquí se describe el sistema desarrollado para resolver el problema, el capítulo se centra en el proceso de desarrollo seguido para alcanzar

la solución, describiendo cada uno de los pasos seguidos y las decisiones tomadas.

- **Experimentación y evaluación:** en este capítulo se demuestra la validez de la solución planteada. Para ello, se muestran los resultados obtenidos como consecuencia de la comparación de las distintas formas de modelar la robustez y valores de los parámetros.
- **Conclusiones y líneas futuras:** en este capítulo se recogen las conclusiones obtenidas tras el desarrollo completo del proyecto, y qué mejoras tienen cabida en el mismo. Se sentarán además las bases para el desarrollo de trabajos futuros y se hará una revisión de los problemas encontrados en el desarrollo del proyecto.

Además, se han incluido una serie de anexos que recogen el seguimiento que se ha hecho del proyecto (incluyendo tanto la planificación inicial como la real), el manual de usuario que acompaña a la aplicación, y un glosario de términos financieros.

Capítulo 2

Contexto

2. ESTUDIO DEL PROBLEMA

En este capítulo se presenta el contexto del problema que este proyecto trata de solucionar, presentando una visión global y amplia sobre el tema. Se describirán los algoritmos genéticos multiobjetivo que se han empleado para la implementación del proyecto, y se mencionarán varias aproximaciones al problema que se ha realizado usando otras técnicas.

Además se llevará a cabo una revisión de las posibles tecnologías a utilizar en la solución del problema y se finalizará definiendo las limitaciones a las que tendremos que hacer frente.

2.1 EL CONTEXTO DEL PROBLEMA

La Teoría Moderna de Cartera o Teoría de Selección de Carteras (*Modern Portfolio Theory*) se originó en un artículo publicado por Harry M. Markowitz ¹, en 1952 [6],[7]. La teoría explica cómo utilizar la diversificación racional para optimizar una cartera. Para ello propone al inversor abordar la cartera como un todo, estudiando las características de riesgo y retorno global, en lugar de escoger valores individuales en virtud del retorno esperado de cada valor en particular. Desde la publicación de este modelo, el diseño y selección de carteras de inversión óptimas ha constituido un problema de gran interés en el área de las finanzas cuantitativas. Un inversor racional tratará de escoger las acciones más prometedoras de acuerdo a ciertos criterios, invirtiendo en diferentes activos cuyos comportamientos están sujetos a incertidumbre. El objetivo es maximizar el rendimiento de la cartera minimizando el riesgo asociado con la inversión, diversificando las inversiones para disminuir el riesgo.

En el modelo de Markowitz, el precio de mercado de los activos es una variable aleatoria cuyos rendimientos son caracterizados por valores de su media y de su varianza. La media es una medida del rendimiento esperado. La varianza es una medida del riesgo asociado a una inversión en dicho valor. El objetivo es calcular el conjunto de combinaciones de rendimiento promedio esperado y del valor esperado de la varianza de dicho rendimiento que son óptimas (frente de Pareto).

Formalmente, Markowitz define estos dos objetivos de la forma que sigue:

- 1) La rentabilidad de cualquier título o cartera, es una variable aleatoria de carácter subjetivo, cuya distribución de probabilidad para el período de referencia es conocido por el inversor. El valor medio (esperanza matemática) se acepta como medida de la rentabilidad de la inversión
- 2) Se acepta como medida del riesgo la dispersión, medida por la varianza o desviación estándar, de la variable aleatoria que describe la rentabilidad, de un valor individual o de una cartera.
- 3) La conducta del inversor le lleva a preferir aquellas carteras con mayor rentabilidad y menor riesgo.

Su modelo de Media-Varianza se define:

- 1) Maximizando el rendimiento o rentabilidad.
- 2) Minimizando el riesgo.

1) La rentabilidad de una cartera será igual a la suma ponderada de las rentabilidades de los activos que la componen. Se ponderarán las rentabilidades por el peso específico que cada activo tiene en la cartera, tal y como se expresa en la siguiente ecuación:

$$E(R_p) = \sum_{i=1}^n w_i \cdot E(R_i) = w_1 \cdot E(R_1) + w_2 \cdot E(R_2) + \dots + w_n \cdot E(R_n)$$

Donde:

$E(R_p)$ = Rentabilidad esperada de la cartera.

W_n = Porcentaje de la cartera (en tanto por uno) invertido en cada acción.

$E(R_i)$ = Rentabilidad esperada de cada acción que entra en la cartera.

2) Convencionalmente, se suele utilizar como medida del riesgo la variabilidad en la tasa de los rendimientos que se obtienen de la inversión, medida por **la desviación típica**. Es decir, el riesgo o volatilidad de un activo financiero se mide por la dispersión de sus posibles resultados. Una distribución con rendimientos sumamente volátiles de un período, indica un alto grado de incertidumbre acerca del rendimiento posible de la inversión (o alto riesgo). Por tanto, cuanto mayor sea la varianza o desviación estándar de la distribución de probabilidades de los posibles rendimientos de una inversión, menos atractiva resultará [8].

La fórmula que define el riesgo o varianza de una cartera de inversiones, que esta función objetivo implementa para intentar minimizar, sería [9]:

$$\sigma_p^2 = \sum_i w_i^2 \sigma_i^2 + \sum_i \sum_{j \neq i} w_i w_j \sigma_i \sigma_j \rho_{ij},$$

Donde ρ_{ij} es el coeficiente de correlación entre los rendimientos de los activos i y j . Alternativamente, la expresión puede expresarse como:

$$\sigma_p^2 = \sum_i \sum_j w_i w_j \sigma_i \sigma_j \rho_{ij}$$

siendo $\rho_{ij} = 1$, cuando $i=j$.

Cuantitativamente, el riesgo se representa con la covarianza, o con la desviación estándar (raíz cuadrada de la varianza):

$$\sigma_p = \sqrt{\sigma_p^2}$$

Como se puede ver, la desviación estándar de una cartera depende no sólo de las varianzas de los instrumentos financieros individuales, sino también de la covarianza de los rendimientos posibles entre cada par de instrumentos financieros.

La covarianza indicará cuál será el comportamiento de un activo al producirse una variación en el valor de otro activo, definiéndose de la siguiente manera [10]:

$$Cov_{ab} = \frac{1}{n-1} \cdot \sum_{i=1}^n (R_{ai} - E[R_a]) \cdot (R_{bi} - E[R_b])$$

Donde R_{ai} y R_{bi} son los posibles valores de rentabilidad para los activos a y b respectivamente.

La covarianza indica en qué medida varía una acción respecto a la otra. De esta forma, si la covarianza es positiva, quiere decir que cuando una acción sube, la otra también tiende a subir; si la covarianza es negativa, quiere decir que cuando “a” sube “b” tiende a bajar. Si la covarianza es próxima a cero, quiere decir que las dos acciones no están relacionadas.

Un parámetro estadístico que también indica la relación entre dos acciones, y que es más fácil de interpretar, es el coeficiente de correlación ρ . Este coeficiente se define por medio de la siguiente ecuación:

$$\rho_{ab} = \frac{Cov_{ab}}{\sigma_a \cdot \sigma_b}$$

Se tiene que:

$$-1 \leq \rho_{ab} \leq 1$$

Al igual que la interpretación de la covarianza, el factor de correlación será positivo si ambas acciones se mueven en el mismo sentido, y será negativo si las acciones se mueven en sentidos opuestos. Por otro lado, si las acciones no tienen ninguna relación entre sí, ρ_{ab} estará en torno a cero.

La ventaja de este coeficiente es que además de poder interpretar el sentido en el cual se mueven ambas acciones, nos entrega información acerca de la magnitud de esta relación, la cual se expresa de la siguiente manera:

- Cercano a 0 “relación entre las acciones débil”
- Cercano a |0,5| “relación entre las acciones moderada”
- Cercano a |1| “relación entre las acciones fuerte”.

El problema, tal y como fue planteado inicialmente por Markowitz, tiene la ventaja de que se puede resolver mediante programación cuadrática. Por otro lado, presenta el inconveniente de que modela una realidad que no se ajusta bien a las condiciones reales de mercado, por lo que se suele extender. El tipo de nuevas restricciones que deben ser consideradas incluyen: umbrales mínimo y máximo de inversión, restricciones en el número

de subyacentes distintos que pueden ser incluidos en la cartera, así como otro tipo de restricciones que quedarían fuera de este estudio (concentración de capital, costes de transacción, costes asociados con la modificación de la composición de la cartera, etc.). Estas modificaciones convierten la selección de carteras en un problema complicado para cuya solución es necesario hacer uso de técnicas más sofisticadas.

Otra limitación que presenta el modelo es el de la robustez de las soluciones. Tal y como ha quedado patente, el cálculo de las rentabilidades y riesgos de las carteras parte de dos conjuntos de parámetros. Por un lado, los rendimientos previstos para los distintos activos y, por otro, la matriz de varianzas-covarianzas que los relaciona. Dado que estos parámetros son desconocidos, tienen que ser estimados y, por tanto, existe la posibilidad de que no se ajusten a la realidad. Habitualmente, estas estimaciones suelen coincidir con la media de los rendimientos históricos y la matriz de varianzas-covarianzas estimadas sobre esos mismos datos. Si la las carteras incluidas en el frente fuesen muy sensibles a este error de estimación, la solución del problema, tal cual suele ser planteado, podría tener una utilidad muy limitada. Es por esta razón por la que en este proyecto se pretende prestar atención a la robustez de las soluciones frente a estos errores.

En este proyecto se pretende alcanzar una mayor robustez de dos formas distintas y potencialmente complementarias: la inclusión de un tercer objetivo explícito de robustez al evaluar las carteras, y el uso de técnicas de remuestreo.

Algunos investigadores definen la robustez como la consistencia de los resultados entre diferentes ejecuciones, o la fiabilidad de éstos en entornos inciertos, donde la función de fitness óptima es variable en el tiempo. [10]

Nuestra intención es utilizar la robustez para **minimizar** la distancia que podría existir entre el par rentabilidad-riesgo de los parámetros obtenidos de los datos históricos, y el que se obtendría al producirse alguna variación imprevista.

Para calcular estas variaciones, se introducen en el sistema dos parámetros de distorsión:

- El **radio** o rango de variación entre las carteras que se están evaluando y las resultantes al introducir ruido.
- El **exponente** al que se elevan estas distorsiones, para penalizar, en mayor medida, las que tengan un valor más alto.

A continuación se calculará la distancia de **Mahalanobis** entre la cartera original y cada una de las resultantes al introducir ruido sobre ella. La métrica es el promedio de todas estas distancias. Se considera que el sistema se está evaluando bajo una situación de normalidad en el mercado, por lo que se espera que las variaciones en él sean mínimas. Por este motivo, se minimizan los resultados obtenidos.

La distancia de Mahalanobis es una distancia que permite cuantificar el grado de semejanza entre dos individuos (en función de sus pares rentabilidad-riesgo). Se diferencia de la distancia euclídea en que tiene en cuenta la correlación entre las variables aleatorias (los individuos de la población inicial).

Formalmente, la distancia de Mahalanobis entre dos variables aleatorias con la misma distribución de probabilidad \vec{x} e \vec{y} , y con matriz de covarianza Σ se define como:

$$d_m(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T \Sigma^{-1} (\vec{x} - \vec{y})}.$$

Donde $(\vec{x} - \vec{y})$ constituye el vector de pares de diferencias rentabilidad-riesgo entre el individuo centroide (el original que se está evaluando) y las distorsiones introducidas sobre él (con un rango de variación r). Σ^{-1} representa la inversa de la matriz de covarianzas, obtenida a partir del producto de un conjunto de individuos aleatorios y las series de rendimientos mensuales de los datos de entrada.

El hecho de tratar más de un objetivo en conflicto, por un lado se busca la mayor rentabilidad y por otra parte el menor riesgo, generará una zona de negociación que permitirá encontrar el óptimo de inversión dependiendo de la tolerancia al riesgo que tengan los inversores. Para resolver este conflicto se recurrirá a la optimización multiobjetivo. [11]

Una gran proporción de los problemas de optimización del mundo real son multiobjetivo. Estos problemas consideran varios objetivos que generalmente están en conflicto y además deben ser satisfechos al mismo tiempo. Desde hace pocos años la computación bio-inspirada, en general, y las técnicas evolutivas en particular, se están aplicando en la optimización de soluciones a problemas de este tipo. La computación evolutiva es un conjunto de algoritmos que presentan una serie de características que permiten explorar espacios de búsqueda de soluciones óptimas, utilizando principios biológicos, cuya eficacia y eficiencia ha sido aplicada y probada en diversos problemas. [12]

2.2 EL ESTADO DE LA CUESTIÓN

El objetivo de esta sección es poner al lector en el contexto de realización del trabajo. Para ello se presenta una revisión, acompañada de una descripción comparativa, de los principales algoritmos, tecnologías y herramientas utilizadas en el desarrollo de este proyecto, con el fin de conseguir una buena optimización de carteras que cumpla con todas las expectativas propuestas.

En la vida real se dan una gran cantidad de situaciones en las cuales el ser humano se encuentra en la disyuntiva de tener que elegir entre diferentes alternativas. Muchas de esas situaciones son habituales y escasamente complejas, por lo que se pueden resolver directamente siguiendo procesos racionales relativamente sencillos. Sin embargo, existen otros problemas de mayor complejidad que no se pueden resolver de forma óptima siguiendo dichos procesos racionales, por lo que es necesario llevar a cabo una planificación formal para tomar la decisión óptima.

En matemáticas, la optimización es la disciplina encargada de encontrar entradas a una función que minimice o maximice su valor, en muchos casos sujetas a restricciones. En la actualidad es posible resolver problemas de la vida real que en el pasado eran intratables gracias a los avances tecnológicos en algorítmica y en hardware de computación. La optimización computacional incluye las disciplinas de investigación operativa para modelar

el sistema, matemáticas para formular el modelo, ciencias de la computación para el diseño y análisis de algoritmos, e ingeniería del software para implementar el modelo.

Para explicar las diferentes técnicas que se han empleado para resolver los problemas de optimización de carteras, se empezará definiendo en qué consiste la optimización. Optimizar es encontrar la solución óptima de entre un conjunto finito de soluciones alternativas. Mientras que en el contexto mono-objetivo la calidad de una determinada solución viene dada directamente por el valor de aptitud (fitness) de la función objetivo. En el contexto multiobjetivo no existe un solo óptimo global, sino varios que optimizan todos los objetivos del problema. En el presente apartado se describen los conceptos necesarios para tratar problemas multiobjetivo, describiendo formalmente la optimización basada en frentes de Pareto, para su posterior utilización. Además, se ofrece una breve descripción de las principales técnicas heurísticas multiobjetivo encontradas en la literatura, clasificándolas en función de la generación a la que pertenecen (primera o segunda). [5]

2.2.1 ELEMENTOS DE OPTIMIZACIÓN

2.2.1.1 PROBLEMAS DE OPTIMIZACIÓN

Los problemas de optimización se caracterizan por [13]:

1. **Variables de decisión**, que son una abstracción de los criterios relevantes del problema y que se representan como cantidades numéricas, las que serán determinadas por la optimización.
2. **Funciones objetivo**, que son las relaciones que deben cumplir las variables de decisión y son expresadas como funciones computables.
3. **Restricciones**, que son las cotas dentro de las que se puede asignar valor a las variables de decisión para que la solución sea factible.

Para los problemas de optimización llamados de Programación Lineal existen métodos de solución como el Simplex; para los de Programación No Lineal se requiere usualmente que la función objetivo sea diferenciable; y en lo general no se conoce ningún algoritmo que permita encontrar la solución óptima para un problema en tiempo polinomial [14].

La optimización global puede entenderse como un problema de hallar un mínimo (o máximo) global. Aunque los problemas de optimización con solo un objetivo pueden tener una única solución óptima, los problemas de optimización multiobjetivo (MOP) presentan posiblemente una incontable combinación de soluciones, las cuales se pueden ver como vectores de un conjunto de puntos dispersos en el espacio de soluciones, de donde el tomador de decisiones seleccionará según sus propios intereses a los vectores que considere como soluciones aceptables.

2.2.1.2 PROBLEMAS DE OPTIMIZACIÓN MULTI OBJETIVO

La denominada optimización vectorial, multicriterio o multiobjetivo (MOP), según Osyczka se define como [14]:

“El problema de encontrar un vector de variables de decisión que satisfaga ciertas restricciones y optimice una función vectorial cuyos elementos representen las funciones objetivo. Estas funciones forman una descripción matemática de los criterios de desempeño que usualmente están en conflicto entre sí y que se suelen medir en unidades diferentes. Por lo tanto, el término optimizar significa encontrar una solución tal que proporcione valores para todos los objetivos que resulten aceptables para el diseñador”.

Hay tres posibles problemas de optimización multiobjetivo:

- Minimizar todas las funciones objetivo.
- Maximizar todas las funciones objetivo.
- Minimizar algunas funciones objetivo y maximizar otras.

Por simplicidad, normalmente todas las funciones son convertidas a una misma forma, a fin de minimizar o maximizar todas las funciones objetivo del problema.

En su forma general, este tipo de problemas pueden ser expresados matemáticamente como [13]:

$$\begin{array}{ll} \text{Optimizar} & y = f(x) = \{f_1(x), f_2(x), \dots, f_k(x)\} \\ \text{Sujeto a} & g(x) = \{g_1(x), g_2(x), \dots, g_k(x)\} \geq 0 \\ & h(x) = \{h_1(x), h_2(x), \dots, h_k(x)\} = 0 \\ \text{donde} & x = \{x_1, x_2, \dots, x_n\} \in X \\ & y = \{y_1, y_2, \dots, y_n\} \in Y \end{array}$$

Donde “ x ” representa el vector de decisión, “ X ” denota el espacio de decisión, “ y ” representa el vector objetivo y el espacio objetivo es denotado por “ Y ”. Al conjunto de todos los vectores de decisión “ x ” que satisfacen las $m+p$ restricciones se le conoce como Conjunto de soluciones factibles y se denota por “ X_f ”. Su imagen se conoce como Región Factible del Espacio Objetivo y se denota por “ Y_f ”. Por lo tanto, el problema consiste en encontrar la “ x ” que proporcione el mejor valor para “ $f(x)$ ”. Sin embargo, la optimización multiobjetivo, no se restringe a la búsqueda de una única solución, sino de un conjunto de ellas llamadas soluciones no-dominadas.

En el año de 1896, *Vilfredo Pareto* estableció el origen de la investigación en optimización multiobjetivo enunciando el concepto de **óptimo de Pareto**: una solución \mathbf{X}^* se dice que es Pareto-óptima si y sólo si no existe otro vector \mathbf{X} tal que $v = f(x) = (v_1, \dots, v_k)$ domine a $u = f(x^*) = (u_1, \dots, u_k)$.

La definición anterior dice que el punto \mathbf{X}^* es un óptimo de Pareto si no existe un vector \mathbf{X} que haga mejorar alguno de los objetivos —respecto a los valores obtenidos para \mathbf{X}^* — sin que empeore de forma simultánea alguno de los otros. En general, la solución

en el sentido de Pareto al problema de optimización multiobjetivo no será única: la solución estará formada por el conjunto de todos los vectores no-dominados, a los que se conoce con el nombre de conjunto de no-dominados o **frente de Pareto** [15].

Los algoritmos de optimización multiobjetivo, tienen por finalidad encontrar un conjunto de soluciones eficientes bajo el concepto de **dominancia de Pareto**. Esta definición permite establecer cuándo una solución, del universo de posibles soluciones, es mejor que otra. La dominancia de Pareto definida para dos vectores de decisión, asumiendo minimización, $x, y \in T$ (T se refiere a la región factible), indica lo siguiente:

Un vector $x = (x_1, x_2, \dots, x_k)$ se dice que domina (en el sentido de Pareto) a otro vector $y = (y_1, y_2, \dots, y_k)$ (que se denota como $x \prec y$) si y sólo si:

$$\forall i \in (1, \dots, k), x_i \leq y_i \wedge \exists i \in (1, \dots, k) : x_i < y_i$$

En otras palabras, un vector domina a otro, cuando éste es menor o igual para todos los componentes y es estrictamente menor en al menos uno de ellos. Es importante notar que si una solución x no domina a otra solución y , e y no domina a x , entonces ambos son no-dominados, y por lo tanto son parte de las soluciones que se buscan a éste problema. En el caso especial de soluciones no-dominadas que se encuentran en los límites de inferiores (para la minimización) de la región factible se conoce como frente de Pareto global.

Resulta imposible determinar de manera analítica la expresión matemática que corresponde al frente de Pareto de un problema arbitrario. Aproximar dicho frente es precisamente el objetivo principal de la optimización de los algoritmos de optimización multiobjetivo.

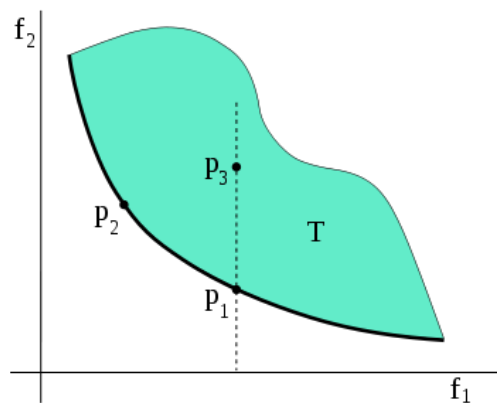


Figura 2: Frente de Pareto de una función con dos objetivos

En la figura 2 se representa, con trazo grueso, el frente de Pareto de una función con dos objetivos. El área coloreada T representa la imagen de dicha función objetivo. Se puede observar que no existe ningún punto perteneciente a T que mejore en el sentido de Pareto, a algún punto del Frente: eligiendo un punto de T de forma arbitraria, por ejemplo p_3 , se puede trazar la vertical hasta obtener el punto de corte con el Frente de Pareto, en este caso p_1 ; dicho punto de corte siempre tendrá el mismo valor de f_1 y un valor mejor de f_2 . También se puede observar que para dos puntos cualesquiera del frente de Pareto, nunca habrá uno que mejore de forma simultánea los dos objetivos respecto al otro punto.

Cogiendo por ejemplo los puntos p_1 y p_2 , se observa que para p_1 mejora f_2 , pero a costa de empeorar f_1 (se está considerando un caso de minimización).

Las diversas técnicas existentes para obtener el frente de Pareto se pueden clasificar en tres categorías: enumerativas, deterministas y estocásticas. En los últimos años, los métodos estocásticos han sido ampliamente estudiados; en particular, un gran número de autores han trabajado en la investigación de algoritmos evolutivos. Estos métodos no garantizan la obtención de la solución óptima, pero ofrecen soluciones aceptables para un amplio rango de problemas de optimización en los que los métodos deterministas encuentran dificultades. En el apartado [2.2.4](#) (Optimización multiobjetivo basada en algoritmos evolutivos), se describirá detalladamente el tratamiento de los problemas multiobjetivo mediante estos algoritmos [14].

2.2.2 TÉCNICAS RECIENTES PARA LA OPTIMIZACIÓN DE CARTERAS

Muchos investigadores han intentado atacar el problema de optimizar una cartera de inversión a través de diversas técnicas. A continuación se explicará de manera resumida las técnicas más empleadas para la resolución de este tipo de problemas. [10]

La complejidad para resolver el problema de selección de carteras está relacionada tanto con el tipo de restricciones que contiene, como con el tamaño del problema. El modelo se vuelve más complejo en proporción con el número de restricciones a tratar y el número de activos manejados. Tomando como base el Modelo de Markowitz descrito anteriormente, la solución más simple del problema de optimización se obtiene cuando la restricción de no-negatividad es omitida del modelo (lo que refleja la admisión de ventas en corto plazo). Para este caso, el método de Lagrange resulta muy útil a la hora de encontrar soluciones fácilmente. Pero cuando se aumenta la restricción de no-negatividad al modelo, el problema de Media-Varianza se vuelve más complejo. El problema obtenido entonces, puede ser aún resuelto por algoritmos como la adaptación de Wolfe al método Simplex (Wolfe, 1959).

Cuando el modelo incluye restricciones de comercio o de número máximo de activos en las carteras, el problema se vuelve de tipo entero mixto no-lineal y los algoritmos clásicos ya no son capaces de encontrar el valor óptimo de la solución del problema (Crama & Schyns, 2003). Uno de los investigadores que han tratado de resolver este tipo de modelos es Perold, quien incluyó en su trabajo un conjunto de restricciones mayor al del modelo simple de Markowitz, pero no puso limitaciones en el número de activos del modelo (Perold, 1984). Takehara fue otro investigador que tomó un modelo más complejo que el de Markowitz y desarrolló un algoritmo de punto interior de optimización de carteras (Takehara, 1993). Muy pocos han estado interesados en la aplicación de heurísticas de búsqueda local para la solución del problema de selección de carteras. Loraschi et al. propusieron utilizar algoritmos evolutivos (Loraschi et al., 1995). Chang et al. experimentaron con una variedad de metaheurísticas, incluyendo Temple simulado (*simulated annealing*) en un modelo sin restricciones de comercio (Chang et al., 2000). Mansini y Speranza han realizado varios trabajos sobre selección de carteras con restricciones de cardinalidad y teniendo en cuenta costes mínimos de transacción a través de técnicas heurísticas (Chiodi et al., 2003; Kellerer et al., 2000; Mansini et al., 2004; Manini & Speranza, 1999). Bauer por su parte, demostró algunos de los potenciales de usar procesos evolutivos en la búsqueda de reglas atractivas de comercio (Bauer & Fiz-Gerald, 2000). Fueron Maringer y Kellerer quienes lograron resolver

el problema de optimización de carteras con restricciones de cardinalidad utilizando un algoritmo híbrido de búsqueda local que combina los principios de Temple simulado con estrategias evolutivas (Maringer & Kellerer, 2003). Como conclusiones de su trabajo, encontraron que esta metaheurística arrojaba resultados fiables y eficientes.

Uno de los trabajos de optimización que reúne mayor número de restricciones es el de Crama y Schyns (Crama & Schyns, 2003). Su modelo de problemas complejos de selección de carteras describe la aplicación del algoritmo de *Temple simulado* para la solución del problema. La adición de restricciones adicionales al modelo de Media-Varianza de Markowitz convirtió el modelo en un problema mixto de programación cuadrática. Como los métodos de optimización exactos demostraron ser ineficientes para resolverlo, ellos propusieron la investigación de técnicas heurísticas que llevaran a mejores resultados. Los experimentos computacionales resultantes, demostraron la capacidad de su algoritmo para encontrar soluciones a problemas de tamaño medio en tiempos aceptables. Además, la forma en la que fue construido el algoritmo permite la modificación o sustitución de la función objetivo por alguna otra medida de riesgo, sin tener la necesidad de alterar al algoritmo, por lo que resulta muy versátil al no estar basado en ninguna propiedad restrictiva del modelo.

La Computación Evolutiva, es una de las familias de técnicas más común para la resolución de problemas de optimización (numérica y combinatoria), y de aprendizaje. Se hace uso de ella, principalmente, en problemas con espacios de búsqueda extensos y no lineales, gracias a su capacidad de encontrar soluciones en donde otros métodos no son capaces de encontrarlas en un tiempo razonable. Es por esto que, como se verá, es una técnica fundamental utilizada tanto para optimización de carteras como en otras aplicaciones. [11].

2.2.3 ALGORITMOS EVOLUTIVOS

La Computación Evolutiva es una vertiente de la Inteligencia Artificial que usa como conceptos de inspiración los de la evolución biológica, con la finalidad de resolver problemas de optimización generalmente de muy alta complejidad. Por ello, la Computación Evolutiva interpreta la naturaleza como una gran máquina de resolver problemas y trata de encontrar el origen de dicha potencialidad para utilizarla a la hora de conseguir respuestas. Sus principios se basan en la teoría Neo-Darwiniana¹ de la evolución, sustentada en el principio de supervivencia del individuo más apto que existe en la Naturaleza, tal y como el famoso naturalista

¹ Neo-Darwinismo: teoría sintética de la evolución:

La teoría evolutiva propuesta originalmente por Charles Darwin en combinación con el seleccionismo de August Weismann y la genética de Gregor Mendel, se conoce como el paradigma Neo-Darwiniano.

El neo-darwinismo establece que la historia de la vasta mayoría de la vida en nuestro planeta puede ser explicada a través de unos cuantos procesos estadísticos que actúan en y dentro de las poblaciones y especies: la reproducción, la mutación, la competencia y la selección.

Se basa en cinco premisas:

1. La estabilidad del proceso de reproducción: los organismos engendran organismos similares.
2. El número de individuos que sobrevive en cada generación, y en cada especie, es siempre menor que el número producido inicialmente: nacen más individuos de los que sobrevivirán.
3. En cualquier población ocurren variaciones aleatorias independientes del ambiente, algunas de las cuales serán hereditarias.
4. Las interacciones entre las variaciones al azar y el ambiente, determinan quiénes sobrevivirán y quiénes no. Las variaciones que permiten sobrevivir y reproducirse, se llaman favorables. Esta es la esencia de la selección natural.
5. Tras suficiente tiempo, la selección lleva a la acumulación de cambios que explica la diversidad de los organismos.

Charles Darwin definió en su libro el de 1859, donde describía la Selección Natural o Supervivencia del más Adaptado como el proceso de preservación de las diferencias y variaciones favorables en cada individuo, así como la destrucción de aquellas variaciones perjudiciales.

Originada desde 1930, con el trabajo pionero de W. D. Cannon, la Computación Evolutiva ha dado pie a tres técnicas principales que fueron concebidas de manera totalmente independiente:

- **La programación evolutiva (PE).**
- **Las estrategias evolutivas (EE).**
- **Los algoritmos genéticos (AG).**

Cada una de ellas se desarrolló con diferentes motivaciones, aunque hoy en día suelen aplicarse, en general, a problemas de optimización (numérica y combinatoria) y de aprendizaje.

Los algoritmos evolutivos (AE) implementan el modelo neo-darwinista descrito en forma de un ciclo computacional, tomando la idea de la supervivencia del más apto de un conjunto de individuos codificando la información de cada solución en un vector a modo de un cromosoma. A diferencia de los métodos clásicos de optimización, como los citados anteriormente, el algoritmo evolutivo no escoge una solución inicial, sino un conjunto de soluciones conocidas como población inicial. Genera artificialmente un nuevo conjunto de individuos usando partes del individuo más apto de la generación anterior y, ocasionalmente, una nueva parte es introducida aleatoriamente para obtener diversidad. La convergencia a una buena solución factible se logra al explotar eficientemente la información histórica que se tiene que especular en la nueva búsqueda, puntos con mayor rendimiento esperado en la función objetivo del problema [14].

Un AE se caracteriza por:

1. **Representación.** Estructura de datos que codifica los parámetros (genes) de una posible solución a un problema.
2. **Población.** Conjunto de individuos que representan las variables de decisión de las funciones objetivo del problema. Utilizan la representación como forma de simular cadenas cromosómicas (individuos) con una carga de información genética.
3. **Operadores genéticos.** Son manipuladores o modificadores de la información genética de los individuos. Promueven la diversificación de individuos dentro de una población, lo que se traduce en el ámbito externo como puntos correspondientes a otras regiones y puntos cercanos a regiones prometedoras del espacio de búsqueda, básicamente son:
 - a) **Mutación.** Variación de uno o más alelos del gen. Su aplicación en forma aleatoria a diferentes puntos de la cadena cromosómica produce individuos con pequeñas variaciones con respecto al individuo original.

- b) **Cruce.** Es el cruce de las cadenas cromosómicas de los individuos padre que generan nuevos individuos. Su aplicación en forma selectiva sobre individuos padre permite que prevalezcan características de un progenitor en su descendencia, pero mezcladas con las características de otros buenos padres.
- c) **Selección.** Se basa en el valor que determinan dichos individuos de su función de aptitud (conocida como *fitness*, y nombrada así a lo largo del proyecto), la cual es consecuencia directa de la influencia del individuo en las funciones objetivo del problema, y sirve para determinar su supervivencia en la siguiente generación.

4. Parámetros. Valores que se proveen para simular el proceso evolutivo: tamaño de la población, probabilidad de aplicar un operador genético, etc.

Cada cromosoma corresponde a un individuo de la población. Habitualmente, los cromosomas son representados por cadenas binarias. Dado que un gen es una subsección de un cromosoma, codificará el valor de una sola variable. (Figura 3)

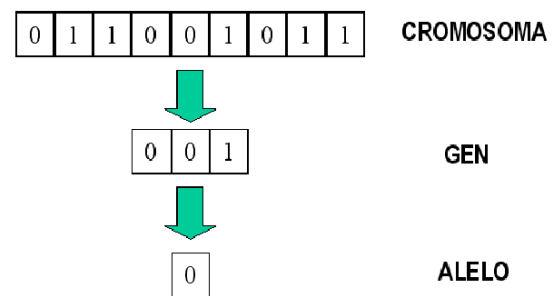


Figura 3: Estructura de un cromosoma

Así pues, el genotipo corresponde con la codificación (por ejemplo, binaria) del cromosoma y el fenotipo con la decodificación de éste. (Figura 4).

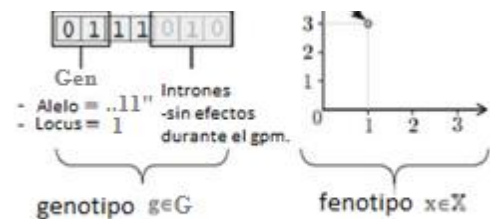


Figura 4: Representación de genotipo y fenotipo

Acorde a la selección natural de los más aptos, en los AE existe un mecanismo que permite que los individuos con mejor fitness se conserven durante el proceso evolutivo sin cruzarse ni mutar. Dicho mecanismo es conocido como elitismo. En general, se fundamenta en retener en la generación *i* de un AE, a los *k* mejores individuos de las últimas *r* generaciones. En su versión más simple consiste en retener (copiar sin ninguna alteración) al mejor individuo de la generación inmediata anterior, lo que asegura que el mejor individuo de la generación *i* + 1 tendrá un fitness, al menos tan alto, como el mejor de la generación *i*, evitando con ello que la mejor solución encontrada hasta el momento sea perdida a causa del proceso evolutivo. Esto asegura el comportamiento monótonico de la mejor función de fitness por generación, lo que es condición necesaria y suficiente para la convergencia del AE.

La evolución es, por lo tanto, el resultado de estos procesos estocásticos fundamentales que interactúan entre sí en las poblaciones, generación tras generación.

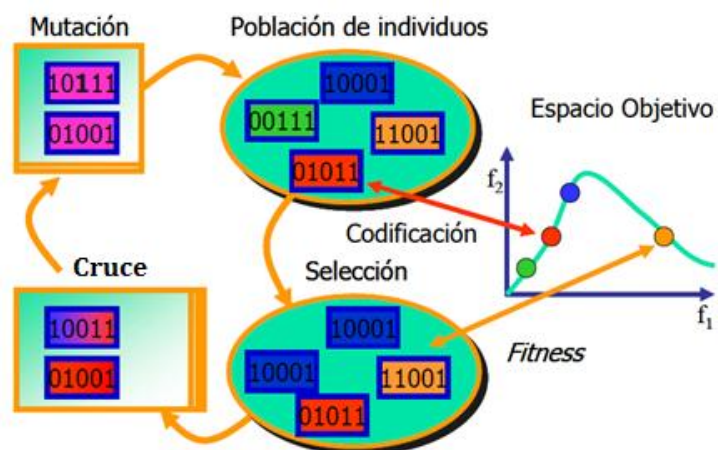


Figura 5: Esquema de la evolución

La diferencia básica existente entre los diferentes enfoques de la computación evolutiva radica en la manera en que los principios evolutivos son utilizados para obtener las características anteriores. Sin pretender analizar en profundidad las diferencias entre los diferentes modelos evolutivos principales, se tiene que:

- Los algoritmos genéticos enfatizan el operador de cruce como el operador de búsqueda más importante y aplican la mutación con una pequeña probabilidad como un operador de segundo plano. Los AG también utilizan un operador de selección probabilístico y normalmente emplean una representación binaria de los individuos.
- Las estrategias evolutivas utilizan mutaciones con distribución normal para modificar vectores de valores reales, enfatizando la mutación y el cruce como sus operadores principales para la exploración del espacio de búsqueda. El operador de selección es determinístico y las poblaciones padre e hijo normalmente difieren en tamaño.
- La programación evolutiva enfatiza la mutación y no incorpora la recombinación de individuos. Al igual que las estrategias evolutivas, utilizan una distribución normal para su operador de mutación. El operador de selección es probabilístico, y aunque en la actualidad la mayor parte de sus aplicaciones son para búsquedas en espacios de vectores de valor real, el algoritmo fue desarrollado en un principio para evolucionar máquinas de estado finito.

De estos tres paradigmas, sólo se describirán, con algo más en detalle, los Algoritmos Genéticos (AG), puesto que los algoritmos multiobjetivos de los que se ha hecho uso para la implementación de este proyecto, utilizan como base este tipo de algoritmos.

2.2.3.1 ALGORITMOS GENÉTICOS

A principios de los años 1960s, John H. Holland, junto con otros colegas y alumnos de la Universidad de Michigan, desarrollaron los AG para resolver problemas de aprendizaje de máquina. Para ello analizaron el fenómeno de adaptación en sistemas naturales y artificiales, logrando abstraer en un primer modelo tentativo, las características esenciales del proceso evolutivo tal como se observa en la naturaleza, con el objeto de utilizarlo en un sistema computacional.

Los AG trabajan con una representación que permite dos interpretaciones, una al nivel del genotipo, que es la carga genética heredada por sus antepasados, y otra al nivel del fenotipo, que son las características visibles del individuo. Los individuos de la población inicial normalmente se generan en forma aleatoria. Los operadores genéticos trabajan a escala genotípica sobre la representación elegida (p. ej., binaria, real, etc.) obteniéndose en consecuencia una nueva generación de la población, donde el operador genético principal es el cruce y el operador secundario es la mutación [14].

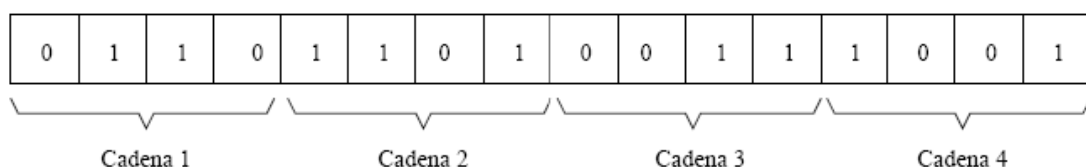


Figura 6: Representación binaria de un individuo

El algoritmo genético se define:

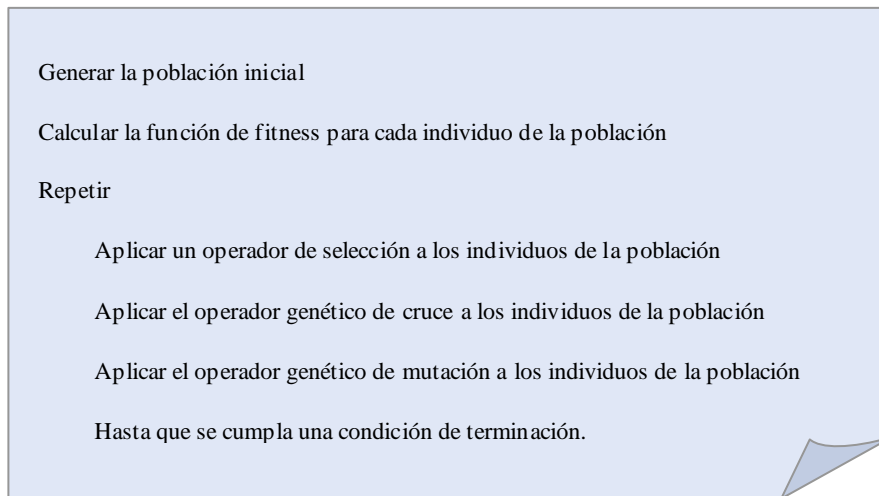


Figura 7: Pseudocódigo de un algoritmo genético

Hoy en día existe toda un área de investigación denominada “Optimización multiobjetivo basada en algoritmos genéticos”. Allí se explotan las ventajas inherentes de los algoritmos genéticos para lograr encontrar en un tiempo razonable un conjunto de soluciones eficientes. Con estos algoritmos es posible encontrar, en una iteración, más de una solución eficiente, por lo que en comparación con las técnicas tradicionales, son capaces de lograr una mejor frontera en un menor tiempo de simulación. Por lo anterior, existe en la literatura un número considerable de técnicas evolutivas para la optimización multiobjetivo, las cuales se describirán en el siguiente apartado.

2.2.4 OPTIMIZACIÓN MULTIOBJETIVO BASADA EN ALGORITMOS EVOLUTIVOS

La optimización multiobjetivo es un área de investigación importante debido a que la mayoría de los problemas del mundo real son, por naturaleza, multiobjetivo. Desde este punto de vista multiobjetivo, no existe una definición de óptimo global, como pudiera ser fácilmente entendida en el caso de la optimización con un solo objetivo. Esto se debe a que los problemas multiobjetivo suelen tener múltiples soluciones óptimas, o un conjunto de soluciones de compromiso alternativas, debido a la naturaleza conflictiva de sus objetivos. Lo que complica la decisión acerca de qué técnica obtiene la mejor respuesta, e incluso el realizar una comparación entre tales técnicas, porque la decisión de cuál es la mejor respuesta la establece el usuario del modelo, quien aplicará para ello sus propios criterios subjetivos [14].

El área de investigación nombrada optimización evolutiva multiobjetivo, se basa en la búsqueda de soluciones utilizando un AE, concretamente algoritmos genéticos, como una forma alternativa para lidiar con los problemas que presentan varios objetivos a la vez.

Rosenberg [16] en el año de 1967, sugiere el primer uso de AG para optimización multiobjetivo, utilizando las propiedades de cercanía a alguna composición química, en su

simulación de la genética y la química de una población de organismos unicelulares. Su implementación consideró una sola propiedad, por lo que la técnica para lidiar con objetivos múltiples no se llevó a la práctica, aunque la sugerencia marcó el inicio de la investigación en torno a optimización multiobjetivo usando técnicas evolutivas. Desde entonces han aparecido en la literatura otros enfoques distintos que se basan en la eficiencia de Pareto y en otros tipos de ordenación, tal y como se puede ver por ejemplo en Fonseca y Fleming (1993).

Si se parte de que un AG requiere de información escalar sobre el valor de la función de fitness de un individuo para operar, la idea más simple que se propone para lidiar con varios objetivos es reunirlos en un objetivo único, usando una combinación de operaciones aritméticas (suma, multiplicación, etc.). No obstante, existen problemas con esta técnica. Primero se debe poseer información precisa sobre el rango de los objetivos, a fin de escalarlos y evitar que uno de ellos domine a los demás en magnitud numérica. Esto implica conocer “a priori” el comportamiento de cada una de las funciones objetivo, lo que en la mayoría de las aplicaciones del mundo real implica un proceso muy costoso (en cuanto a recursos computacionales de los datos y sus medidas estadísticas). Si esta combinación de objetivos es posible, esta técnica no sólo es la más simple de implementar, sino que, además es la más eficiente, porque no se requiere posterior interacción con el usuario, y si el AG tiene éxito en el proceso de optimización, entonces los resultados serán, al menos, subóptimos en la mayoría de los casos.

La primera implementación de un algoritmo evolutivo multiobjetivo fue realizada por Schaffer en la década de los ochenta con el Vector Evaluated Genetic Algorithm (VEGA). Con ello inicia la primera generación de AEMO que se caracterizan por el uso de mecanismos de selección basados en el concepto de optimalidad a través de la dominancia de Pareto, y por el uso de diferentes técnicas para la compartición de fitness para mantener diversidad (nichos, σ_{share}). Los algoritmos más representativos de esta primera generación fueron: el Nondominated Sorting Genetic Algorithm (NSGA) de 1994, el Niche-Pareto Genetic Algorithm (NPGA) de 1994, y el Multi-Objective Genetic Algorithm (MOGA) de 1993.

La implementación del mecanismo de elitismo en el contexto de optimización multiobjetivo dio paso a la segunda generación de algoritmos evolutivos multiobjetivo. El elitismo en este contexto normalmente es implementado a través de una población externa o secundaria, en la que se almacena a los individuos no-dominados encontrados durante el proceso de búsqueda. También puede ser implementado mediante el uso de selección ($\mu + \lambda$), donde los padres compiten contra los hijos, y los individuos no-dominados resultantes son los que se retienen para la siguiente generación. Sin embargo, en cualquiera de los dos casos, se requiere cumplir con restricciones que ayuden a lograr la mejor distribución posible de los individuos no-dominados. Los principales algoritmos de esta segunda generación son: el Strength Pareto Evolutionary Algorithm (SPEA) de 1999, el Strength Pareto Evolutionary Algorithm 2 (SPEA2) de 2001, el Pareto Archived Evolution Strategy (PAES) de 2000, el Nondominated Sorting Genetic Algorithm II (NSGA-II) de 2000, el Niche-Pareto Genetic Algorithm 2 (NPGA2) de 2001, el Pareto Enveloped-based Selection Algorithm (PESA) de 2000, y el Micro Genetic Algorithm (micro-GA) de 2001.

De las diferentes técnicas evolutivas que se han propuesto para la optimización multiobjetivo, para los fines de este trabajo se consideraron los dos representativos del estado del arte en el área: NSGA-II, y SPEA2 [4],[5],[13],[17],[18]. Interesa evaluar

aquellos métodos basados en óptimos de Pareto y, sobre todo, aquellos métodos que corresponden a los desarrollados en la segunda generación.

A continuación se describirán, de manera general, estos dos algoritmos seleccionados, para su posterior comparación en la fase de experimentación.

NSGA-II

En 1994, N. Srinivas y Kalyanmoy Deb propusieron el NSGA (Nondominated Sorting Genetic Algorithm) que utiliza una idea de jerarquización de soluciones con base en dominancia de Pareto propuesta por Goldberg, la cual se realiza por ondas o capas (waves). Se comparte aptitud entre los individuos con el fin de mantener la diversidad.

La segunda versión de este algoritmo, llamada NSGA-II fue propuesta por Deb, Agrawal, Pratap y Meyarivan en el año 2000, con el fin de incorporar elitismo y reducir la complejidad del procedimiento de ordenación rápido por no dominancia de su antecesor (NSGA). Realiza una clasificación de la población por frentes. Los individuos que pertenecen al primer frente son los no dominados; los que pertenecen al segundo frente son los no dominados en ausencia de los del frente anterior, y así sucesivamente. A cada individuo se le asigna un rango equivalente a su nivel de no dominancia. Los mejores individuos son aquellos que tienen rangos menores. También incorpora el cálculo de una distancia de *crowding*, como el operador utilizado para mantener la diversidad de la población, con el fin de evitar el uso del (σ_{share} , nichos) en la compartición del fitness (*fitness sharing*) de su antecesor. La selección es realizada mediante torneo binario, utilizando como criterio de comparación el operador \succ_n . Según este criterio, el torneo lo gana el individuo con menor rango. Si el rango es el mismo, el torneo lo gana aquel individuo que tenga menor distancia de *crowding*.

Sin embargo, el NSGA-II muestra problemas al generar regiones aisladas del frente de Pareto, así como para lidiar con más de dos funciones objetivo [19].

SPEA2:

Desarrollado por Zitzler, Laumanns y Thiele en el 2000 con el fin de superar debilidades detectadas en el esquema de asignación de adaptación del SPEA. En este algoritmo, la función de fitness se mejora teniendo en cuenta para cada individuo el número de individuos a los que domina y el número de individuos por los que es dominado. Este esquema también añade una estimación de densidad poblacional. El tamaño – N_{EMax} – de la población externa P_E (utilizada para elitismo) es fijo, a diferencia del SPEA, en el cual el tamaño de P_E es variable pero acotado. P_E está conformada sólo por individuos no dominados, siempre y cuando el número de éstos sea mayor o igual que N_{EMax} . En el caso en que el número de individuos no dominados sea menor que N_{EMax} se incluyen individuos dominados dentro de P_E hasta que el tamaño de P_E sea igual a N_{EMax} . La técnica de agrupamiento (*clustering*), encargada de mantener la diversidad de la población en SPEA, es sustituida por un método de truncamiento, el cual evita eliminar las soluciones extremas del conjunto de soluciones no dominadas. La selección se realiza mediante torneo binario, tomando como criterio de comparación el fitness de cada uno de los individuos. SPEA2

asume minimización de fitness, por lo tanto gana el torneo aquel individuo que tenga un menor valor de fitness.

En general, los parámetros utilizados por NSGA-II y SPEA2 son los siguientes: tamaño de la población, tamaño de la población élite, número máximo de generaciones (g_{\max}), número máximo de generaciones de convergencia (g_{conv}), probabilidad de cruce (P_{cruce}) y probabilidad de mutación ($P_{\text{mutación}}$).

2.2.5. APLICACIONES DE LOS ALGORITMOS GENÉTICOS MULTIOBJETIVO

Son muchos los campos en los que estos algoritmos proporcionan una gran utilidad, siendo cuantiosas sus aplicaciones dentro de cada uno de ellos. Entre dichos campos, cabría destacar la Medicina (Planificación de tratamientos, Modelos de pronósticos, Reconstrucción en 3D del ventrículo izquierdo, etc.), Ecología, Ciencias de la Computación (citar todas sus posibles aplicaciones en este campo nos alargaría el estudio enormemente), Diseño y Manufactura, *Scheduling*, Física, Química, Geografía Clasificación y Predicción, Ingeniería Aeronáutica, Aplicaciones Científicas, Ingeniería estructural y Mecánica, Ingeniería Eléctrica y Electrónica, Telecomunicaciones y Redes, Ingeniería Ambiental, Naval e Hidráulica, Economía y Finanzas (dominio de este estudio), etc. Aunque el volumen actual de aplicaciones es inmenso, muchos dominios no han sido abordados todavía. Por ejemplo: visión por computadora, ajuste de modelos basados en elementos finitos, diseño de formas, control de *bloat* en programación genética, reconocimiento de patrones, más problemas de optimización combinatoria, etc. [20].

2.2.6 ALGORITMOS GENÉTICOS MULTIOBJETIVO EN LA OPTIMIZACIÓN DE CARTERAS DE INVERSIÓN

En los trabajos revisados: [4], [5], [12], [14], [21], [22], se aplican variantes del modelo de Media-Varianza de Markowitz como: manejo de ventas a crédito, administración de carteras de préstamos o créditos, administración de carteras eficientes, etc. En este proyecto se tendrán en cuenta las restricciones de cardinalidad, así como los umbrales de compra mínimos y máximos para cada activo.

Además, solamente los AG de segunda generación: NSGA-II, SPEA2 y variantes de ES, han sido aplicados a la solución del modelo de Media-Varianza; y son pocas las comparaciones de desempeño realizadas entre ellos, de los cuales ninguno de ellos incluye la aportación de la robustez de la solución entre sus objetivos. El propósito de este proyecto es llevar a la práctica estas carencias encontradas, mediante una comparativa entre dos de los algoritmos más conocidos de segunda generación (NSGA-II y SPEA2), tal y como se ha venido explicando, previa implementación de una cartera de inversiones óptima robusta.

2.2.7 HERRAMIENTAS PARA LA IMPLEMENTACIÓN DE ALGORITMOS EVOLUTIVOS MULTIOBJETIVO

Existen un gran número de herramientas que facilitan la optimización de problemas multiobjetivo mediante:

- Implementación de un gran número de algoritmos.
- Implementación de un gran número de problemas reales y test.
- Implementación de indicadores de calidad.
- Posibilidad de validación estadística.

Algunos ejemplos son: *PISA* (ETH, Suiza), *Paradiseo.MOEO* (INRIA, Francia), *jMetal* (UMA, Málaga),..., etc. En el apartado [4.1.3](#) (Plataforma y herramientas de desarrollo), se analizarán las ventajas y desventajas de algunos de ellos, y los motivos por los que se eligió el *framework jMetal*, y *Eclipse* como entorno de trabajo para la implementación del sistema desarrollado en este proyecto.

Capítulo 3

Gestión del Proyecto Software

3. GESTIÓN DE PROYECTO SOFTWARE

Se entiende por gestión de proyecto al proceso de planteamiento, ejecución y control de un proyecto, desde su comienzo hasta su conclusión, con el fin de alcanzar el mejor resultado posible en cuanto a coste, plazos y calidad [23]. En los siguientes apartados se analizarán en detalle estas cuestiones y se incluirá un análisis de los riesgos que pudieran surgir durante la realización del proyecto y cómo hacerles frente. Este capítulo está planteado tal y como se gestionaría el proyecto en un caso real. Aunque se trata pues de una gestión ficticia, donde algunos conceptos cuyo coste se detalla, en realidad no suponen gasto económico alguno, se describen los gastos que supondría realizarlos en un entorno empresarial. El objetivo es demostrar los conocimientos adquiridos a lo largo de la carrera en cuanto a gestión de proyectos se refiere.

3.1 PLAN DE TRABAJO

Antes de acotar el alcance del proyecto y el presupuesto que supone su realización, es necesario identificar las tareas en las que éste se va a dividir para hacer una estimación de su duración y poder llevar a cabo una planificación adecuada de las mismas.

3.1.1 IDENTIFICACIÓN DE TAREAS

Una vez avanzada la naturaleza del proyecto (claramente se corresponde a un estudio perteneciente a la rama de Inteligencia Artificial), se ha considerado apropiado dividirlo en las siguientes tareas:

- **Identificación del problema:** esta tarea comienza con la detección del problema que nuestro programa pretende resolver, para continuar con la fase de documentación del dominio y comparativa de frameworks. En esta fase conviene realizar una estimación de la planificación, que se deberá seguir durante su desarrollo. Por lo tanto, podría dividirse en tres partes:
 - **Estudio preliminar del dominio del problema:** al tratarse de un tema del que no se tiene conocimiento, como es el de las Finanzas, y más concretamente el de las acciones y carteras de inversión, una previa documentación ayudará a entender el problema y la finalidad que se pretende alcanzar con la implementación del proyecto.
 - **Elaboración del plan del proyecto:** tras detectar el problema y plantear una posible solución, es necesario decidir cómo se va a gestionar el tiempo y los recursos disponibles para su realización. Es por tanto en esta fase, cuando se detalla la planificación a seguir, las tareas en que se dividirá el proyecto y cuál será el coste que supondrá su desarrollo completo.

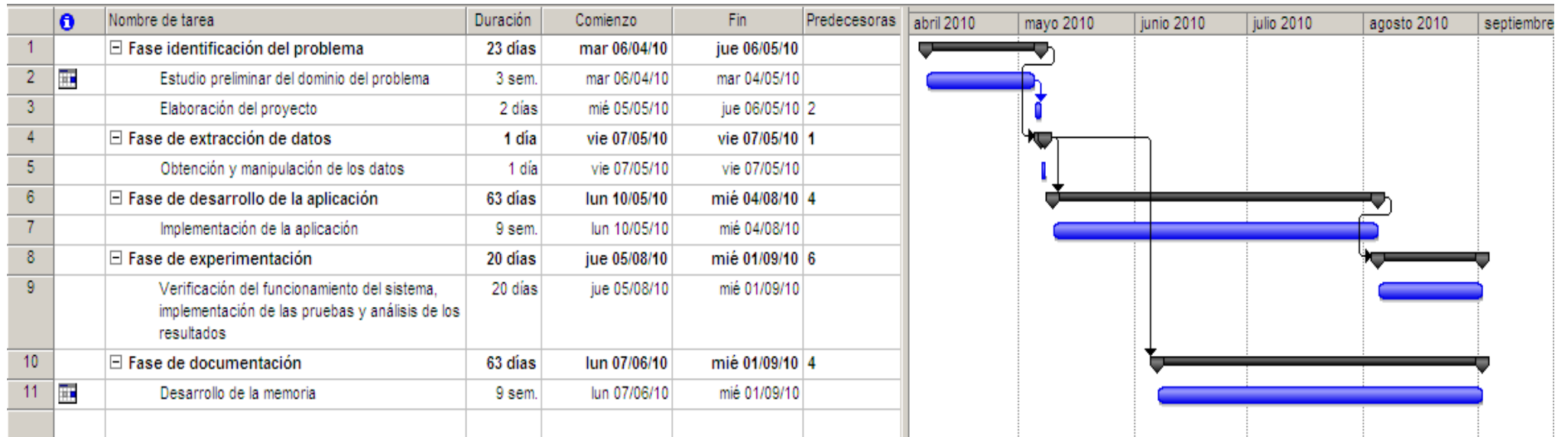
- **Extracción de datos:** esta fase se descompone en dos tareas: la obtención y manipulación de los datos sobre los que trabajará el algoritmo genético multiobjetivo. Estos datos fueron facilitados por el tutor, por lo que mi trabajo quedó reducido a su comprensión y conversión de valores de índices a series de rendimientos, mediante la aplicación de una fórmula sencilla definida en Excel. Posteriormente hubo que cambiarlos de fichero y extensión para poder leerlos desde la aplicación.
- **Desarrollo de la aplicación:** implementación del algoritmo que permita la optimización de carteras de inversión con los objetivos marcados en la definición del problema. Constituye la fase más larga. Podría dividirse en tantas tareas como objetivos pretende optimizar el programa.
- **Estudio exploratorio:** el objeto de esto es establecer los parámetros adecuados para la ejecución de los algoritmos genéticos multiobjetivo, y esperar a su finalización. Puede dividirse en dos tareas:
 - **Estudio exploratorio con el algoritmo NSGA-II:** se ejecutará el programa haciendo uso de este algoritmo para la obtención de resultados que verifiquen por qué se ha centrado el estudio con el algoritmo siguiente, SPEA2.
 - **Estudio exploratorio con el algoritmo SPEA2:** de manera análoga a como se ha descrito en la tarea anterior, se ejecutará la aplicación haciendo uso de este algoritmo para la obtención de los resultados.
- **Verificación de los resultados:** esta fase consiste en comprobar el buen funcionamiento del programa. Es decir, en comprobar no solo que los resultados obtenidos sean correctos, sino también la calidad de los mismos (si sería posible que éstos estuviesen, incluso, a la altura de la elección de un experto en el dominio).
- **Experimentación:** en esta fase se recogen muestras de los resultados obtenidos, y se estudia, con la ayuda de contrastes estadísticos, tablas y gráficas, bajo qué parámetros y número de objetivos (incluyendo o no la estabilidad entre ellos), se obtienen los mejores resultados para el algoritmo SPEA2.
- **Documentación:** esta fase consiste en la generación del presente documento, incluyendo desde la explicación de en qué consiste el proyecto, hasta las gráficas que muestren el resultado de la experimentación y el manual de usuario.

3.1.2 ESTIMACIÓN DE TAREAS

Una vez definidas las tareas en que se va a dividir el proyecto, es necesario estimar los días que va a llevar cada una de las tareas y fases, con el fin de permitir hacerse una idea más precisa de la envergadura del proyecto. La duración estimada para cada una de las tareas puede verse detallada en la siguiente tabla:

Fase	Tarea	Días estimados
Fase identificación del problema	Estudio preliminar del dominio del problema	21 días
	Elaboración del plan del proyecto	2 días
	Total fase identificación del problema	23 días
Fase de extracción de datos	Obtención y manipulación de datos	1 día
	Total fase de extracción de datos	1 día
Fase de desarrollo de la aplicación	Implementación de la aplicación	63 días
	Total fase desarrollo de la aplicación	63 días
Fase de experimentación	Verificación del funcionamiento del sistema, implementación de las pruebas y análisis de los resultados	20 días
	Total fase de experimentación	20 días
Fase de documentación	Desarrollo de la memoria	60 días
	Total fase de documentación	60 días

Tabla 1: Descripción estimada por tareas y fases



3.2 ALCANCE DEL PROYECTO

Una vez identificadas las tareas y definido el plan de trabajo, el siguiente paso a la hora de gestionar un proyecto es definir su alcance, es decir, delimitar el trabajo que se ha de realizar para cumplir con los objetivos y desarrollar las tareas que se han de ejecutar. En esta sección se describirán los límites del proyecto, definiendo los recursos que serán necesarios para llevarlo a buen término.

3.2.1 DEFINICIÓN DEL PROYECTO

El objetivo de este proyecto consiste en explorar métodos para optimizar carteras de inversiones de forma robusta. Para conseguirlo, se hará uso de algoritmos genéticos multiobjetivo que resuelvan el modelo de selección de inversiones planteado por Markowitz como una optimización multiobjetivo. Se maximiza la rentabilidad y se minimizan el riesgo y la sensibilidad a los errores de estimación de los parámetros del modelo. Para resolver este problema se recurrirá a un algoritmo genético de optimización multiobjetivo, basado en criterios de dominancia de Pareto. Concretamente se utilizará el algoritmo *Strength Pareto Evolutionary Algorithm* (SPEA2), que ya ha demostrado su eficacia en problemas relacionados.

3.2.2 ESTIMACIÓN DE TAREAS Y RECURSOS

En esta sección se van a detallar los costes que supondrán el proceso de análisis y desarrollo del proyecto, desglosándolos en recursos humanos, equipamiento, consumibles, gastos adicionales, etc. Para ello, se tendrá en cuenta que el tiempo que durará este proyecto serán unos cuatro meses: últimas tres semanas de abril, mayo, junio, primera quincena de julio, segunda de agosto y primera semana de septiembre, por lo que todos los gastos se detallarán en función de esa cantidad de tiempo.

3.2.2.1. RECURSOS HUMANOS

En primer lugar se desglosarán los gastos referentes al personal implicado en la realización de este proyecto, debido a su importancia dentro del desarrollo del mismo. El equipo de trabajo está compuesto por 2 miembros, cada uno de ellos con un puesto diferente y un coste por hora dependiente del mismo. Los costes de personal incluyen los honorarios de la estudiante en Ingeniería Informática encargada del desarrollo del proyecto, que en este capítulo adquirirá el rol de Ingeniera, pese a no tener aún el título, y del experto en Inteligencia Artificial y Finanzas (concretamente es Economista), que le asesora.

Se toma como valor orientativo que los honorarios de un Ingeniero Informático especializado en Inteligencia Artificial, sin experiencia, es de 20 €/hora. Teniendo en cuenta que se estima realizar el proyecto¹ en cuatro meses, a un ritmo de trabajo más o menos constante, y a unas nueve horas de media diarias, se calcula que se invertirán unas 1080 horas de trabajo durante dicho período. Por tanto, el coste asociado a dicho concepto asciende a 21.600 euros (veintiún mil seiscientos euros).

¹ En realidad ninguno de los participantes (ni el tutor, ni la desarrolladora del PFC, que ha asumido el rol de

Se estima que los honorarios de los expertos que asesoran al ingeniero desarrollador ascienden a 35€/hora. Se realizarán varias consultas puntuales a los mismos y se calcula que, en total, se requerirán 50 horas de trabajo por parte de los expertos. Por tanto, el coste asociado a este concepto asciende a 1.750 euros (mil setecientos cincuenta euros).

Teniendo en cuenta ambos aspectos, el coste total debido a gastos en personal asciende a 23.350 euros (veintitrés mil trescientos cincuenta euros).

Concepto	Trabajo realizado	Honorarios	Importe
Ingeniero Informático	1080 horas	20 €/hora	21.600€
Experto en IA y Finanzas	50 horas	35€	1.750€
Total gastos personal			23.350€

Tabla 2: Desglose del coste por hora según el rol del trabajador

Tras estimar los gastos en personal es necesario analizar los gastos que va a suponer el equipamiento para el desarrollo del proyecto. Estos gastos se dividen en tres grupos, hardware, software y gastos asociados al presupuesto de trabajo.

3.2.2.2. GASTOS DE HARDWARE

Los gastos de hardware necesarios para el desarrollo de la aplicación incluyen simplemente un ordenador que permita el desarrollo de la aplicación. A condición de tratar de disminuir los tiempos de ejecución, de considerable duración debido al gran número de ellas que se pretenden realizar, es preferible un ordenador con suficiente potencia y RAM, que eviten una ejecución pesada. Por tanto, ha de considerarse el uso de un ordenador portátil *Sony Vaio*, con un procesador *Core 2 Duo T9600 a 2.8 GHz- Centrino 2-* con 8 GB de RAM, comprado con vistas al desarrollo del proyecto, valorado en 2000 euros, con el seguro a tres años incluido. Para calcular el precio que se invertirá en el ordenador para la realización del proyecto, se tendrá en cuenta la amortización del período que se empleará para su desarrollo, ya que al no ser su uso exclusivamente para este fin, no sería correcto considerar todo su importe como gasto del proyecto. Así pues, partiendo de un período de amortización a cuatro años de vida útil, se obtendría una amortización mensual de $2000€/48 = 41.67€$. Considerando que en el proyecto se invertirán cuatro meses, el coste total en este aspecto sería: $4 * 41.67 € = 166.68€$ (ciento sesenta y seis euros y sesenta y ocho céntimos).

Concepto	€/unidad	Nº unidades	Importe
Ordenador portátil	166.68€	1	166.68 €
Total			166.68 €

Tabla 3: Desglose del coste en Hardware.

3.2.2.3. GASTOS DE SOFTWARE

Son necesarios varios productos *software* para la implementación del proyecto. Aunque en su mayoría se emplean con carácter *open source* (código libre, y por consiguiente de distribución gratuita), siendo estrictamente realistas habría que incluir en los gastos la compra cuatro licencias: el sistema operativo (Windows 7 Home Premium), el paquete *MS Office*, *MS Project*, y la herramienta *Altova UModel 2008*. Respecto a esta última herramienta, con precio de 189€ y necesaria para el modelado de clases, se empleará una versión de prueba de 90 días, una vez finalizado el código (tiempo más que suficiente para poder utilizarla gratuitamente). Por otra parte, el precio del sistema operativo (119,99 US\$ = 97,32 euros) estaría incluido junto con el ordenador, pues ya venía instalado cuando se efectuó la compra. Y, respecto a las otras dos licencias, tampoco serían necesarias puesto que, como miembro de la Universidad Carlos III de Madrid, me ha sido facilitado el acceso a una licencia especial que Microsoft tiene con la universidad, llamada “*MSDN Academy*” y que permite la descarga gratuita de algunos de sus productos; y ambos paquetes (*Ms Office Professional 2007* (≈652,00 €) y *Ms Project 2007 Professional* (≈666.15€)) fueron obtenidos por medio de esta licencia.

Los recursos *open source* a utilizar son: el framework que se empleará para la implementación del código (*JMetal*), el entorno de desarrollo *Eclipse*, *JavaEE* como lenguaje de programación, y varias librerías de *Commons-Math* (proporcionadas por *Apache Commons*, implementadas en *Java*) para facilitar el desarrollo de ciertas funciones.

Si se quisiera comercializar con el producto final fruto del proyecto, mencionar que al recurrir al empleo de *JMetal* para su implementación, con licencia *LGPL* (es decir, bajo licencia de *Creative Commons GNU Lesser General Public License*), no habría ningún problema, siempre y cuando se distribuyeran los cambios efectuados al propio framework. En otras palabras, cualquier modificación en su código implicaría la distribución gratuita del código fuente del framework, con mis modificaciones realizadas, a quien lo solicitase.

Finalmente, se considerarán como gastos de software la compra de las licencias de las herramientas *MS Office*, *Ms Project* y *Altova UModel*, pese a no necesitarse invertir nada en este apartado. Así pues, los gastos derivados de este punto sumarían un total de 1507.15€ (mil quinientos siete euros y quince céntimos), cuyo desglose quedaría:

Licencia	€/unidad	Nº unidades	Importe
Ms Office	652 €	1	652 €
Ms Project	66,15 €	1	666,15 €
Altova UModel 2008	189 €	1	189 €
Total			1.507,15 €

Tabla 4: Desglose del coste en Software.

3.2.2.4. OTROS GASTOS

Para elaborar el presupuesto final, también hay que tener en cuenta los gastos asociados con el desarrollo del proyecto, tales como los consumibles y gastos de comunicaciones, que ascienden a un total de 232,80 € (doscientos treinta y dos euros y ochenta céntimos):

Concepto	Descripción	Importe
Folios Tinta	Folios para la documentación	32,80 €
	Costes de impresión y encuadernación	100 €
Gastos comunicación (Internet)	Precio de la línea ADSL * 4 meses	100 €
Total		232, 80 €

Tabla 5: Desglose de los costes asociados con el desarrollo del proyecto

Para ser más realistas, en el cálculo del presupuesto final del proyecto se incluyen además otros gastos, detallados a continuación:

- **Riesgos:** se ha añadido al presupuesto final un 10% extra para hacer frente a los posibles riesgos que pudieran surgir durante el desarrollo del proyecto.
- **Beneficios:** se ha incrementado el presupuesto en un 10%, que representan los beneficios tras la finalización del proyecto.
- **IVA:** se ha considerado además, el 18% de IVA reglamentario.

Así, el presupuesto total sin I.V.A del presente proyecto asciende a 30.307,95 € (treinta mil trescientos siete euros y noventa y cinco céntimos). El presupuesto final, incluyendo el I.V.A, y que constituye el total que se deberá pagar por él, asciende a **35.763,38 € (TREINTA Y CINCO MIL SETECIENTOS SESENTA Y TRES EUROS Y TREINTA Y OCHO CÉNTIMOS)**. El desglose completo del presupuesto final puede apreciarse en la siguiente tabla:

Descripción	Importe
Gastos de personal	23.350 €
Gastos hardware	166,68 €
Gastos software	1.507,15 €
Gastos derivados	232,80 €
Total costes directos	25.256,63 €
Riesgos (10%)	2.525,66 €
Beneficios (10%)	2.525,66 €
Total sin I.V.A	30.307,95 €
IVA (18%)	54.55,43 €
Total a pagar	35.763,38 €

Tabla 6: Presupuesto total del proyecto

Capítulo 4

Desarrollo del proyecto

4. DESARROLLO DEL PROYECTO

En este capítulo se explica el proceso de diseño e implementación del algoritmo desarrollado, que garantiza la viabilidad y la obtención de la cartera de inversiones óptima. Para lograrlo se han empleado, de la forma más adecuada y reutilizable posible, las técnicas explicadas en el apartado [2.24](#) (Optimización multiobjetivo basada en algoritmos evolutivos).

Así, a partir de los elementos que componen un AE, y utilizando como modelo los algoritmos evolutivos multiobjetivo, también comentados en el capítulo de contexto ([2.1 Contexto del problema](#)), se realiza la implementación de la solución evolutiva. Ésta se inicia con el diseño del cromosoma que se utilizará en los algoritmos evolutivos multiobjetivo. A continuación se analiza el tratamiento a aplicar a la representación del problema, así como a los individuos obtenidos tras el cruce y mutación, y se especifican las características de los problemas a simular. Para ello se establecen los parámetros requeridos para la ejecución de cada uno de los dos algoritmos, con los que más tarde se experimentará y realizarán comparaciones.

4.1 ANÁLISIS Y DISEÑO DE LA SOLUCIÓN

Tal y como se ha explicado, el objetivo del proyecto es evaluar el uso del algoritmo evolutivo multiobjetivo, SPEA2, en el problema de selección de instrumentos financieros que conformarán carteras eficientes de inversión. Para esto, se establecerá el porcentaje a invertir de cada instrumento financiero utilizando el modelo de Análisis de Riesgo-Beneficio definido por Markowitz, aplicado de tal modo que se minimice la sensibilidad de las soluciones a errores de estimación en los parámetros.

Hasta ahora se ha explicado en qué consiste el problema implementar y las diferentes técnicas y herramientas existentes para poder abordarlo; también se ha calculado el coste que supondría su implementación. En esta sección se detallarán las diferentes decisiones de diseño tomadas y la forma en la que han sido implementadas.

Al no tratarse de un proyecto destinado a realizar ninguna aplicación para un cliente final, sino más relacionado con la investigación sobre un área específica de la Inteligencia Artificial, la parte referida al análisis y el diseño de la aplicación que se ha creado ha sido pensada con total independencia, adecuándose a los objetivos de investigación previamente descritos. Así, la implementación se ha ido desarrollando de una manera estructurada y ordenada, de modo que su complejidad iba incrementándose conforme se ramificaba el estudio por los diversos aspectos de los Algoritmos Genéticos Multiobjetivo y de las Carteras de inversión.

El diseño desarrollado deberá cumplir los siguientes requisitos:

- Es necesario establecer cuáles son los objetivos que han de cumplirse en este proyecto para establecer unas conclusiones.

- Es necesario cumplir cada uno de esos objetivos de manera independiente de cómo se implemente el programa, es decir, la implementación no debe limitar o reducir el ámbito del sistema.
- La implementación debe realizarse de tal forma que se minimice el impacto producido por cualquier modificación posterior en el análisis o diseño. Es requisito para este proyecto que dicho análisis pueda ser modificado añadiendo nuevas funcionalidades o modificando las ya implementadas.
- La implementación debe ser flexible, permitiendo especificar con facilidad los parámetros necesarios para la aplicación y los algoritmos genéticos.

Antes de mostrar el diseño y cometido de las clases e interfaces que componen el sistema implementado, se realizará un análisis de los requisitos y características generales que debe cumplir la aplicación, y que permite justificar la necesidad de la existencia de estas clases.

4.1.2 ANÁLISIS

En este punto se explican las características generales del sistema desarrollado, mediante la enumeración de una serie de requisitos y funcionalidades que se pretenden cumplir con la implementación de la aplicación.

4.1.2.1. CAPACIDADES Y RESTRICCIONES GENERALES:

A) REQUISITOS DE IMPLEMENTACIÓN

Todo algoritmo evolutivo multiobjetivo se define, al menos, mediante estos dos requisitos mínimos:

- 1) La evolución de varias poblaciones soluciones al problema.
- 2) La utilización mecanismos que mantengan diversidad en la población para conseguir un conjunto de soluciones no dominadas para la adaptada.

A continuación se definen una serie de requisitos que debe cumplir la aplicación:

- 1) Debe permitir al usuario seleccionar el tipo de estudio que desea realizar (optimización de carteras con dos objetivos, optimización de carteras incluyendo estabilidad en el sistema, hacer experimentos con la estabilidad y analizar las soluciones reales).
- 2) Deberá ser intuitiva y fácil de manejar. La utilización de un menú permitirá trabajar con ella de manera sencilla, mediante la interacción.

- 3) Permitirá realizar tantas ejecuciones del algoritmo genético como se especifiquen, para obtener muestras con suficientes datos para el análisis. Cada ejecución deberá ser independiente de las anteriores, para conservar todos los frentes de Pareto resultantes y facilitar su representación gráfica y estudio estadístico.
- 4) En cada ejecución podrán utilizarse los algoritmos NSGA-II y SPEA2, simultáneamente, para hacer comparativas entre ambos.
- 5) Podrán añadirse nuevas métricas de desempeño para la evaluación del rendimiento de los algoritmos.
- 6) No deben sobrescribirse los mejores frentes de Pareto ni sus individuos correspondientes, obtenidos en cada ejecución.
- 7) La ejecución de cada prueba generará un directorio con el nombre del experimento realizado. En su interior se creará otra carpeta con el nombre del estudio llevado a cabo para ese experimento. A su vez, para cada estudio, se crearán tres nuevos directorios con los datos obtenidos, y scripts a partir de ellos, para facilitar el análisis estadístico.
 - El subdirectorio “*Data*” almacenará una carpeta por cada algoritmo utilizado. Y para cada algoritmo, se crearán tantos directorios como problemas se hayan ejecutado a partir de ellos. En su interior se incluirán los frentes de Pareto generados y sus individuos, así como las métricas obtenidas a partir de los frentes. Para el análisis de la estabilidad, la última opción del menú, se crearán tanto los frentes de Pareto estimados, como los frentes de Pareto efectivos, además de una métrica diferente a la del resto de los experimentos.
 - El directorio “*R*” almacenará los scripts en lenguaje R, para poder realizar contrastes con ellos.
 - Y el tercer y último subdirectorio (“*Latex*”) contendrá las tablas en Látex con los resultados estadísticos (media, mediana, desviación típica, IQR, máximos y mínimos).
 - Los archivos generados deberán tener un formato interpretable por el lenguaje R, con el objetivo de facilitar la realización de contrastes estadísticos durante la fase de análisis.
- 8) Podrán explotarse fácilmente arquitecturas en paralelo, permitiendo la optimización de varios problemas simultáneamente.

B) REQUISITOS DE FUNCIONALIDAD

En esta sección se explican las características generales del sistema desarrollado. Esto comprende las funcionalidades que ofrece la aplicación, así como ciertas características propias del problema a implementar que deben cumplirse.

En primer lugar, las funciones objetivo implementadas debe cumplir el siguiente requisito que cumpla con la Teoría de Cartera descrita en el punto [2.1](#) (El contexto del problema):

- Dado un nivel de riesgo, debe obtenerse el máximo rendimiento (o, dicho de manera opuesta, dado un nivel de rendimiento debe obtenerse el menor riesgo).

Este requerimiento es independiente a la necesidad de implementar otras funciones objetivo, además del riesgo y del rendimiento (para el estudio que nos ocupa la tercera función sería la robustez), y de otras restricciones además de las propias del modelo de Markowitz.

A su vez, esta aplicación debe ser extensible y modificable, adaptándose a la estructura y filosofía del framework utilizado para su implementación. Así, en base a los resultados obtenidos, posibles cambios en las funciones objetivo y en las restricciones, o ampliaciones en el número de ambas, podrían hacerse efectivos en la aplicación de una forma sencilla.

Con este objetivo, la aplicación debe mostrar las siguientes funcionalidades:

1. Debe implementar los objetivos mencionados anteriormente: maximizar el rendimiento y minimizar el riesgo y la sensibilidad a los errores de estimación en los parámetros del modelo de Markowitz.
2. Permitirá especificar los parámetros del problema de optimización de carteras y de los algoritmos genéticos de los que se esté haciendo uso, de una forma sencilla y flexible.

Concretamente, esto se controlará a través de un fichero de entrada, en el que se permitirá especificar los siguientes parámetros:

- El modo en que se ejecutará la aplicación, en función de las series de rendimientos mensuales que se seleccionen: modo estático (sin remuestreo) o aleatorio (con remuestreo).
- El número máximo de activos permitidos para la definición de la restricción de cardinalidad.
- La definición de los umbrales máximos y mínimos de compra para cada uno de los activos. Los valores de estos umbrales son independientes para cada activo.
- El número máximo de activos que compondrán la cartera a optimizar. El programa deberá ser lo suficiente robusto para permitir cualquier número de ellos.
- El número máximo de meses que se considerarán para calcular el riesgo, rendimiento y estabilidad de las series de rendimientos de las que se dispone.
- El número máximo de variantes a generar de los individuos cuya estabilidad se pretende evaluar.

- La magnitud $[-r, r]$ de las modificaciones que se aplican sobre las variables de los individuos cuya estabilidad se pretende analizar..
- El exponente al que se elevará la distancia de Mahalanobis con el objetivo de penalizar los individuos más inestables.

Los parámetros de los algoritmos genéticos deben poder modificarse, bien interactuando con el menú, o bien por medio de un archivo de configuración. Estos parámetros son:

- El tamaño de la población, que coincidirá con el del frente de Pareto resultante de la ejecución.
 - El número máximo de generaciones que evaluarán al individuo.
 - Las probabilidades de los operadores genéticos de cruce y mutación.
 - Los índices de distribución del cruce y de la mutación.
3. Los valores de los parámetros relacionados con el problema de las carteras de inversión se especificarán mediante un fichero de entrada, y los parámetros de los algoritmos genéticos mediante un fichero de configuración (o interactuando con la aplicación).
 4. Se tiene un modo de representación que simula la definición de variables de decisión mediante la existencia de individuos y poblaciones de individuos. Un elemento importante es la diversificación de los individuos, ya que a mayor diversificación se explora un mayor número de regiones del espacio de búsqueda (otras soluciones en cada generación).
 5. Se tiene una función de fitness que permite valorar a los individuos y su influencia en las funciones objetivo establecidas, lo que encauza a los nuevos individuos hacia la búsqueda de óptimos con respecto a las funciones objetivo.
 6. Una vez finalizada la ejecución, debe mostrarse el tiempo que se ha empleado para la obtención del frente de Pareto.
 7. La ejecución devolverá dos ficheros: uno con los porcentajes a invertir en cada activo de los mejores individuos. Y otro con los valores de riesgo, rendimiento y distancias de Mahalanobis correspondientes a dichos porcentajes.
 8. No presentará limitaciones en cuanto al número de activos que componen la cartera, tamaño de población o número de individuos que rodean a los que se están evaluando. Estas limitaciones vendrán dadas por la memoria del sistema en el que se ejecute la aplicación.

Una vez establecidos los requisitos y funcionamiento básicos de la aplicación, y definidas las características de los algoritmos genéticos multiobjetivo sobre los que se asienta el sistema, el siguiente paso será explicar las herramientas que se han empleado para su implementación y la plataforma sobre la que se ha llevado a cabo.

4.1.3 PLATAFORMA Y HERRAMIENTAS DE DESARROLLO

Se propusieron dos *frameworks* para facilitar la implementación del proyecto. A continuación se analizarán las características de ambos y los motivos que nos llevaron a la elección del framework utilizado. Tras este análisis, se describirán los posibles entornos de programación con los que llevar a cabo la implementación del sistema a desarrollar.

Antes de comenzar, aclarar que un *framework* es una librería potente que se usa como base para desarrollar aplicaciones. Normalmente contiene una arquitectura bien definida, así como un conjunto de componentes reusables diseñados para simplificar el proceso de desarrollo, aumentar la consistencia y productividad y mejorar la calidad final de la aplicación. Permite a los diseñadores centrarse en el problema a resolver, olvidándose de los detalles de implementación de más bajo nivel.

Los frameworks propuestos son:

- **ParadisEO:** es un framework de caja blanca orientado a objetos que se emplea para el diseño de distintos tipos de metaheurísticas: híbridas, distribuidas y paralelas. Además proporciona un amplio abanico de propiedades como: algoritmos evolutivos, búsquedas locales, optimización de enjambres de partículas, los mecanismos más comunes de hibridación y los modelos más estandarizados de paralelización y distribución, etc. Este alto contenido y utilidad fomentan su uso a nivel internacional. ParadisEO está basado en una clara separación conceptual entre los métodos de solución y los problemas que dichos métodos tratan de resolver. Esta separación confiere al usuario la máxima reusabilidad de código y diseño. Más allá, la baja granularidad de las clases del framework proporcionan una alta flexibilidad comparado con otros existentes. Se trata de uno de los únicos capaces de proporcionar los modelos más comunes de paralelización y distribución. Su implementación es portable en máquinas de memoria distribuida, así como en multiprocesadores de memoria compartida, ya que utiliza librerías estándar como *MPI*, *PVM* y *PThreads*. Los modelos pueden ser explotados de manera transparente, solo es necesario instanciar las clases asociadas proporcionadas. [24]
- **JMetal:** Su nombre proviene de Java Metaheuristics Algorithms, y es un framework orientado a objetos basado en Java. Enfocado en el desarrollo, experimentación y estudio de las metaheurísticas para resolver problemas de optimización multiobjetivo, JMetal proporciona un completo conjunto de clases que pueden ser utilizadas como los bloques para construir metaheurísticas multiobjetivo. Aprovechándose de la reutilización de código, los algoritmos comparten los mismos componentes base, como puede verse en la implementación de los operadores genéticos y los estimadores de densidad, lo que facilita el desarrollo de nuevos algoritmos multiobjetivo. Implementa varias metaheurísticas multiobjetivo, además de poner a disposición varios de los problemas incluidos en estudios de rendimiento. Permite paralelización para la ejecución de experimentos con varios problemas en paralelo, mediante el uso de hilos. JMetal proporciona también indicadores de calidad para calcular el rendimiento, así como una conjunto de utilidades que ayudan a llevar a cabo estudios experimentales. [25]

Tras haber implementado diversos ejemplos con ambos frameworks, trabajar con JMetal resultó mucho más cómodo. Resultaría poco creíble negar el peso que el lenguaje de programación ejerció a la hora de su decisión. Además, sabiendo de antemano la necesidad de importar ciertas librerías matemáticas para el cálculo de matrices de varianza y covarianza, la buena política de estandarización de librerías llevada a cabo por Java (mediante clases e interfaces) y su facilidad para referenciarlas, influyeron en parte a la hora de su elección.

Este lenguaje de programación fue desarrollado por Sun Microsystems a principios de los años 90. Fue definido dentro del paradigma de la orientación a objetos y con el objetivo de permitir ejecutar un mismo programa en diferentes sistemas operativos [Gosling, 1996].

Para ello, el código se compila en bytecodes que son interpretados por la Máquina Virtual Java (JVM, Java Virtual Machine) [Lindholm, 1999].

Este lenguaje, además de aportar las ventajas de la orientación a objetos y permitir desarrollar aplicaciones bastante portables, proporciona una extensa interfaz de programación de aplicaciones (del inglés API², Application Programming Interface). A su vez, al tratarse de un lenguaje muy utilizado, es posible encontrar un gran número de utilidades y bibliotecas, desarrolladas en este lenguaje para ámbitos diversos.

Con base a la experiencia de uso de Java, mencionar que además de los argumentos anteriores, en su elección influyeron también las habilidades adquiridas en su manejo durante la carrera.

4.1.3.1 ENTORNOS DE PROGRAMACIÓN:

Una vez elegidos el lenguaje y, consecuentemente, el framework con el que trabajar, el entorno no supuso ninguna duda. A pesar de que tanto *Eclipse* como *Netbeans* son dos potentes herramientas que presentan grandes facilidades para la implementación de código en Java, de software libre y frecuentes actualizaciones, *Eclipse* fue finalmente el entorno elegido, especialmente por haber trabajado más con él, por poseer un soporte de refactorización más potente que el de *Netbeans* y por su cualidad de añadir nuevos *plugins* (complementos que aportan nuevas funcionalidades) con tan solo copiarlos a la carpeta indicada para tal fin, además de ser más configurable, rápido en ejecución y consumir menos recursos. Sin embargo no está de más mencionar una funcionalidad que *Eclipse* no explota igualmente, y en la que este proyecto no se veía afectado: el gran editor de Interfaces Java para el uso de los paquetes *java.swing* y *java.awt* del que dispone.

A continuación se muestra el conjunto de clases e interfaces necesarias para la implementación del sistema. Pero antes de comenzar con el diseño de toda la estructura y entramado de clases, se proporcionará un esquema con el funcionamiento general de la aplicación.

² <http://java.sun.com/javase/6/docs/api/>

FUNCIONAMIENTO DE LA APLICACIÓN

Antes de adentrarnos en los detalles específicos del diseño de la aplicación, se muestra un esquema (figura 8) con la descripción, de forma gráfica, de su funcionamiento, para facilitar la comprensión del mismo. En el esquema pueden verse los cálculos previos necesarios para resolver el problema de optimización de una cartera de inversión robusta. Puede observarse cómo se relacionan entre sí las diferentes funciones objetivo a optimizar.

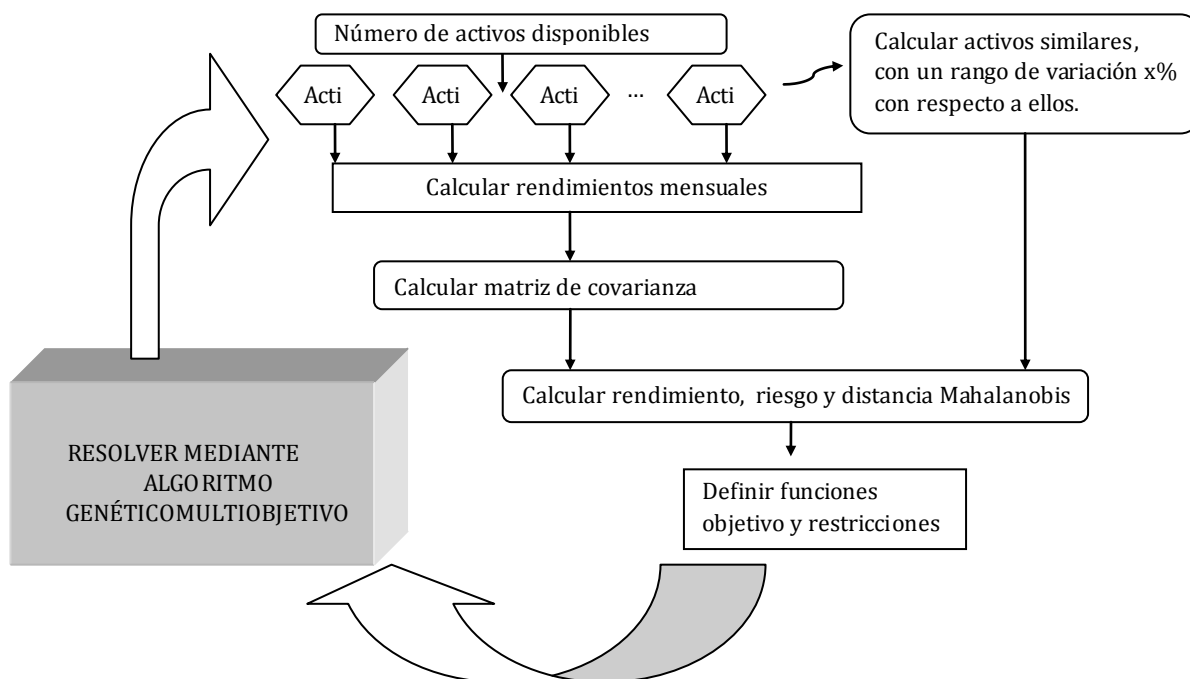


Figura 8: Esquema del funcionamiento de la aplicación

4.1.4 DISEÑO DE LA APLICACIÓN

En esta sección se define la estructura del sistema y se especifican las clases e interfaces de las que consta, así como las relaciones entre ellas. También se detallan los principales métodos y otros detalles relativos a la implementación de la aplicación.

En este diseño se muestran sólo los métodos y funciones principales del sistema, por motivos de extensión y legibilidad.

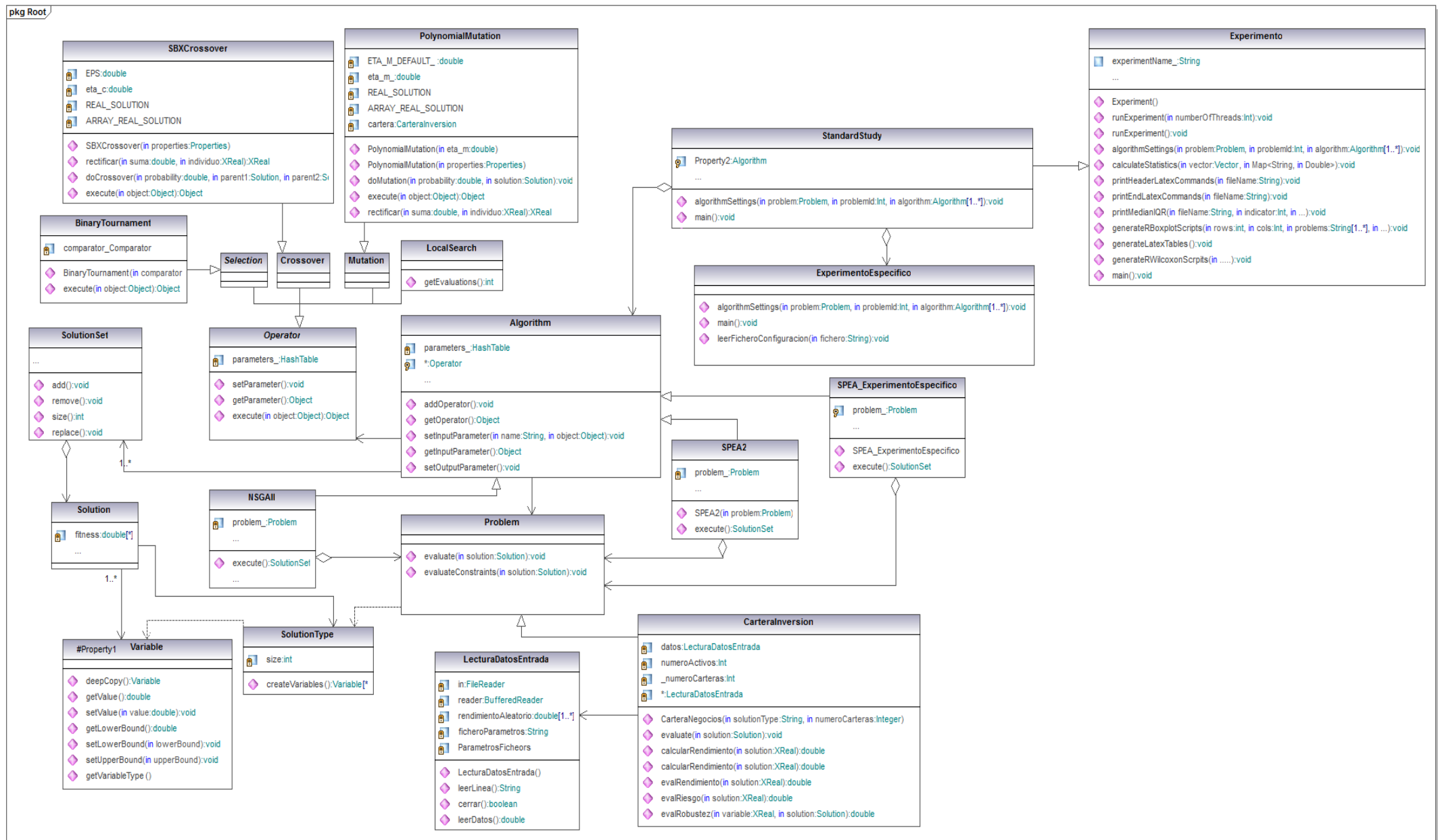


Figura 9: Esquema del funcionamiento de la aplicación

El framework se encuentra encapsulado y estructurado de la forma más adecuada posible, pensando en la reutilización y posibles modificaciones posteriores. A continuación, se llevará a cabo un análisis del funcionamiento del framework, de las distintas clases e interfaces que lo componen. Amoldarse a él para la adición de nuevas clases y funcionalidades no supuso gran dificultad. Esto nos ha permitido centrarnos en el estudio que nos ocupa y conseguir todos los objetivos y requisitos propuestos.

4.1.4.1. CLASE SOLUTION:

Una de las primeras decisiones que hay que tomar cuando se utilizan metaheurísticas es definir cómo se va a codificar o representar las codificaciones provisionales del problema planteado. La representación depende enormemente del problema en particular a resolver, y determina las operaciones que pueden aplicarse (la forma de realizar el cruce, la mutación, etc.). Por lo tanto, la selección de una representación específica tiene un gran impacto en el comportamiento de metaheurísticas y, consecuentemente, en los resultados obtenidos.

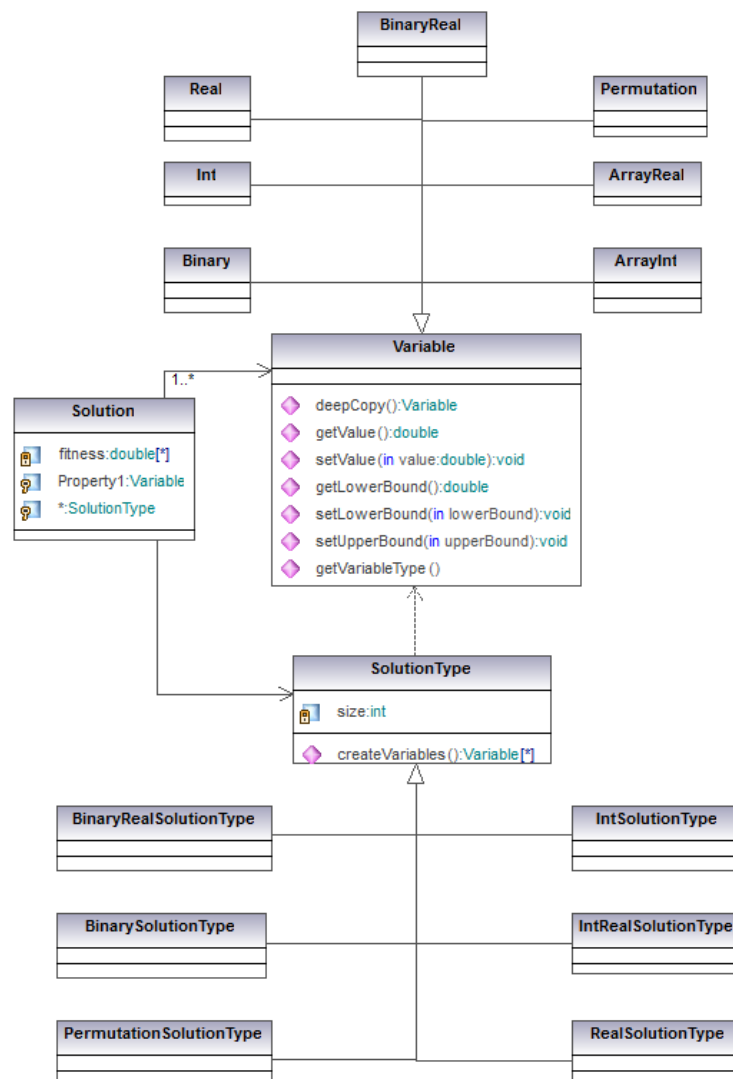


Figura 10: Componentes utilizados para representar la solución

La figura 10 muestra los componentes básicos que se utilizan para representar las soluciones (individuos) en el framework. Una “Solution” (solución), está compuesta por un conjunto de objetos de tipo “Variable”, que pueden ser de diferentes tipos (binario, real, binario con codificación real, enteros, permutación, etc), además de un array para almacenar los valores de fitness. Con la idea de ofrecer un sistema flexible y extensible, cada solución tiene asociado un tipo (la clase *SolutionType* de la figura). El tipo de solución permite definir tipos de variable de la clase *Solution* y crearlas, usando el método *createVariables()*. Esto se ilustra en la figura 11, donde se muestra la codificación de la clase *ArrayRealSolutionType* (omitiendo detalles irrelevantes), usado para caracterizar soluciones compuestas solo por arrays de variables reales. Existen tipos de soluciones similares para representar individuos con codificación entera, binaria, permutación, y otras representaciones, tal y como se muestra en la figura 10.

Es posible utilizar tipos de soluciones sencillas para definir representaciones más complejas, mezclando diferentes tipos de variables. Por ejemplo, para crear una nueva representación de una solución consistente en un real, entero y permutación de enteros, basta con heredar de la clase *SolutionType*, que puede ser definida para representar un nuevo tipo, donde básicamente sólo el método *createVariables()* debe ser redefinido. La figura 11 muestra el código requerido para este tipo de solución.

```
public class RealSolutionType extends SolutionType {
    // Constructor
    public RealSolutionType(Problem problem) {
        ...
    } // Constructor

    public Variable[] createVariables() {
        Variable[] variables = new
        Variable[problem_.getNumberOfVariables()];
        for (int var = 0; var < problem_.getNumberOfVariables();
        var++)
            variables[var] = new Real();
        return variables ;
    } // createVariables
} // RealSolutionType
```

Figura 11: Código de la clase *RealSolutionType*, para representar variables reales

4.1.4.2. CLASE OPERATOR

Las técnicas metaheurísticas se basan en la modificación o generación de nuevas soluciones a partir de las ya existentes, mediante la aplicación de los diferentes tipos de operadores. Por ejemplo, los algoritmos evolutivos hacen uso de los operadores de cruce, mutación y selección para la modificación de soluciones. En el *framework*, cualquier operación puede alterar o generar soluciones (o conjuntos de ellas), heredando de la clase “Operator”, como puede verse en la figura 9. El *framework* incorpora una serie de operadores, que pueden clasificarse en cuatro clases diferentes:

- Crossover (cruce): representa el operador de cruce utilizado en los algoritmos evolutivos. Algunos de estos operadores incluidos en el *framework* son el cruce de simulación binaria (SBX) y el cruce de dos puntos para codificaciones reales y binarias, respectivamente.
- Mutation (mutación): representa el operador de mutación utilizado en los algoritmos evolutivos. Se incluyen ejemplos de operadores de mutación tales como “la mutación polinomial”, para la codificación real, y “la mutación *bit-flip*” para la codificación binaria.
- Selection (selección): este tipo de operador se utiliza para llevar a cabo procedimientos de selección para los algoritmos evolutivos. Un ejemplo de operador de selección es el torneo binario.
- LocalSearch (búsqueda local): esta clase está diseñada para representar procedimientos de búsqueda local. Contiene un método extra de consulta del número de evaluaciones realizadas después de su aplicación.

Cada operador contiene los métodos *setParameter()* y *getParameter()* empleados para añadir y tener acceso a parámetros específicos del operador. Por ejemplo, el cruce *SBX* requiere dos parámetros: una probabilidad de cruce (como la mayoría de los operadores de cruce), además de un valor para el índice de distribución (específico del operador), mientras que un operador de mutación de un solo punto requiere únicamente la probabilidad de mutación.

Es importante señalar que al aplicar un operador sobre una solución determinada, se conoce de antemano de qué tipo es la solución. Por lo tanto, se puede definir un único operador de cruce de dos puntos, por ejemplo, para aplicarse tanto a las soluciones binarias como a las reales, utilizando el tipo de solución para seleccionar el código adecuado en cada caso.

4.1.4.3. CLASE PROBLEM

Todos los problemas del *framework* heredan de la clase *Problem*. Esta clase contiene dos métodos fundamentales: *evaluate()* y *evaluateConstraints()*. Ambos métodos reciben una *Solution* representando una solución candidata para el problema; el primero de ellos la evalúa y el segundo determina la violación global de restricciones para esa solución. Todos los problemas tienen que definir el método *evaluate()*, mientras que únicamente los problemas que tengan restricciones necesitan redefinir el método *evaluateConstraints()*. En el problema se definen los tipos de soluciones que permiten resolverlo. Cada instancia de un problema tiene un método constructor que recibe una cadena con un identificador del tipo de solución que acepta. Además, se definen las características básicas del problema (número de variables, de objetivos y de restricciones). Después del constructor se redefine el método *evaluate()*. En este método, se calculan los valores de las funciones objetivo, los cuales se almacenan posteriormente en la solución mediante el método *SetObjective()* de la clase “*Solution*”.

4.1.4.4. CLASE ALGORITHM

La última clase principal a destacar en el diagrama de *UML* de la figura 3.1, es la clase *Algorithm*. Se trata de una clase abstracta que debe ser heredada por todas las metaheurísticas del *framework*. En particular, debe implementarse el método abstracto *execute()*. Este método está diseñado para ejecutar el algoritmo, devolviendo como resultado un objeto de tipo *SolutionSet*. Una instancia de la clase *Algorithm* requiere, en primer lugar, el problema concreto a resolver (*algorithm = new SPEA2(problem)*), así como la aplicación de una serie de parámetros específicos del algoritmo en cuestión, que pueden agregarse y accederse mediante los métodos *addParameter()* y *getParameter()*, respectivamente. Estos parámetros son por ejemplo el tamaño de la población inicial o el número de evaluaciones que se asignan al algoritmo: (*algorithm.setInputParameter("populationSize", 100)*). Del mismo modo, un algoritmo también podrá hacer uso de algunos operadores, mediante métodos que permiten añadirlos (*addOperator()*), y obtenerlos (*getOperator()*). En el *framework* se incluye la implementación de una serie de algoritmos genéticos multiobjetivo, siendo los siguientes algunos de los más destacados: SPEA2 (utilizado para este estudio), PAES, OMOPSO, MOCcell, AbYSS, MOEAD/D, GDE2, IBEA, SMPSO, o NSGA-II (también utilizado en este proyecto).

Una vez explicado el cometido principal de las clases más importantes del *framework*, se centrará la atención en la implementación de los diferentes elementos de los algoritmos genéticos que permiten la optimización de la cartera de inversiones: el individuo, los operadores de cruce y mutación, así como la función de fitness (dividida en tres objetivos). También se describirán los parámetros de entrada del fichero que contiene las series de los rendimientos mensuales históricos de los instrumentos financieros que componen la cartera.

4.1.4.5. ALGORITMOS NSGA-II Y SPEA2

A continuación se muestra el pseudocódigo de un algoritmo evolutivo multiobjetivo, acompañado por un esquema con los pseudocódigos de los algoritmos utilizados para la optimización del problema, NSGA-II y SPEA2.

```
Inicializar (P(0))
Generación = 0;
Evaluar (P0)
Mientras (no CriterioParada) hacer
    Operador de diversidad (P (generación))
    Asignar fitness (P (generación))
    Padres = Selección (P (generación))
    Hijos = Operadores de Reproduccion (Padres)
    NuevaPop = Reemplazar (Hijos, P (generación))
    Generación++
    P (generación) = NuevaPop
Evaluar (P (0))
Retornar Mejor Solucion Hallada
```

Figura 12: Esquema de un Algoritmo Evolutivo para Optimización Multiobjetivo

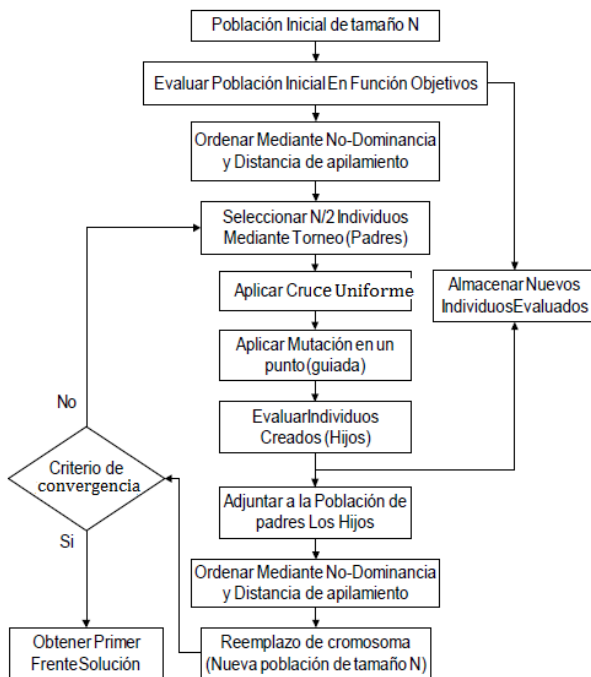


Figura 14: Pseudocódigo del algoritmo NSGA-II

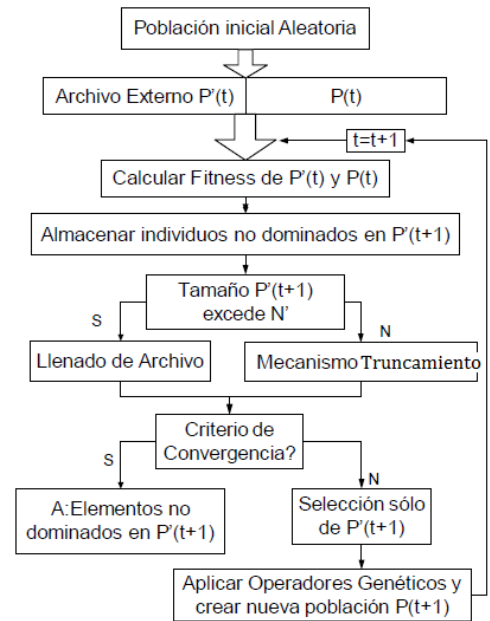


Figura 13: Pseudocódigo del algoritmo SPEA2.

4.1.4.6. CLASE CARTERAINVERSION

En esta clase se implementan las propiedades principales del problema. En primer lugar se dan valor a sus características básicas, tales como el número de objetivos, el número de variables de decisión (en este caso es el número de activos, 8), los límites superiores e inferiores de dichas variables (que coinciden con los umbrales de compra máxima y mínima), se establece el tipo de datos al que pertenecen (arrayReal), y se da nombre al problema. A continuación se define la función de fitness global, encargada de llamar al resto de las funciones objetivo del problema, indicando cuáles se maximizan y cuáles se minimizan (implementación por defecto en el framework). En cada una de ellas, se implementa el cometido a cumplir por cada objetivo.

4.1.4.7. CLASE LECTURADATOSENTRADA

Esta clase es la encargada de leer los parámetros del fichero de entrada, que se explicarán más adelante. Además, en ella se calcula la matriz de varianza-covarianza necesaria para calcular la función objetivo del riesgo, a partir de las series de rendimientos mensuales leídas.

4.1.4.8. INTERFAZ EXPERIMENT

Para la implementación de las pruebas de este proyecto, ha sido muy importante poder hacer uso de esta interfaz. Mediante la implementación de un considerable número de experimentos que heredaban de ella, nos ha permitido automatizar una gran cantidad de ejecuciones independientes entre sí, facilitando, además, su posterior análisis.

La implementación del experimento en su fase inicial era sencilla. Bastaba con crear una clase que heredara de `Experiments`, siguiendo los siguientes pasos:

1. Se configuran los algoritmos con los que se van a realizar los experimentos (en nuestro caso `SPEA2` y `NSGAI`). Se modifica el algoritmo de los archivos correspondientes a `main.java`, o bien se modifican los valores de los parámetros de configuración del objeto asociado.
2. Opcionalmente, se pueden configurar los problemas a resolver, si se quiere trabajar con más de uno (en nuestro caso solo se trabajará con el problema `CarteraInversion`). Para esta opción se trabaja en paralelo, empleando un hilo por problema.
3. Se selecciona el archivo a utilizar como frente de referencia en las pruebas.
4. Se le da nombre al experimento, y se establece un número de ejecuciones independientes (42 en nuestro caso). Y el problema queda listo para su ejecución.
5. Se generan *scripts* para obtener las tablas de `látex` y “`boxplots`” en `R`, que permiten la obtención de información estadística sobre los resultados de las métricas obtenidos. Estos resultados fueron de gran ayuda para el análisis de los experimentos. Sin embargo, no se hizo uso del script en `R`, ya que se implementó otro que se ajustara en mayor medida a nuestros objetivos, como se explicará en la fase de experimentación.

En función del número de objetivos del experimento y, por tanto, de su cometido, podía ser suficiente con la implementación de esta clase. Es decir, para la prueba con solo dos objetivos, utilizada en la selección del algoritmo que mejor se adapte a nuestro problema, bastaba con analizar los resultados obtenidos y decantarse por uno u otro algoritmo.

Sin embargo, para el análisis de la robustez, este experimento era solo el comienzo de varias pruebas a aplicar. Una vez obtenidos los resultados, un fichero con los mejores individuos y otro con los mejores frentes de Pareto, era necesario dividir el tercer objetivo en tres niveles de estabilidad. Para ello, se volvía a llamar a otra clase que heredaba de *Experiment (EstudioEstabilidad)*, que al llamar al algoritmo se encargara de dividir el tercer objetivo en tres niveles distintos, a través de percentiles con valor entre 0.33 y 0.66, los cuales se almacenarán en tres archivos diferentes. En ellos solamente se escribirán los objetivos de riesgo y rendimiento, omitiendo el de la robustez, para la realización de las gráficas y la comparativa con respecto a los frentes de referencia obtenidos con dos objetivos, que no tienen en cuenta la estabilidad.

```

double[][] frente = metrica.readFront("FUN."+ numero);
double estabilidadA = 0.0;
double estabilidadB = 0.0;
double estabilidadInferior[] = new double[500];
double estabilidadMedia[] = new double[500];
double estabilidadSuperior[] = new double[500];
Percentile estabilidadBaja = new Percentile(33.0);
Percentile estabilidadAlta = new Percentile(66.0);
double [] estabilidad = new double[frente.length];
for(int i = 0; i<estabilidad.length; i++){
    estabilidad[i]= frente[i][2];
}
estabilidadA = estabilidadAlta.evaluate(estabilidad, 33.3);
estabilidadB = estabilidadBaja.evaluate(estabilidad, 66.6);

FileWriter file;
try {
    for (int i = 0; i < frente.length; i++) {
        if (frente[i][2] < estabilidadB) {
            file = new FileWriter("EstabilidadSuperior."+ numero, true);
            file.write(frente[i][0] + " " + frente[i][1] + "\n");
        } else if (frente[i][2] > estabilidadB && frente[i][2] < estabilidadA) {
            file = new FileWriter("EstabilidadMedia."+ numero, true);
            file.write(frente[i][0] + " " + frente[i][1] + "\n");
        } else {
            file = new FileWriter("EstabilidadInferior." + numero, true);
            file.write(frente[i][0] + " " + frente[i][1] + "\n");
        }

        file.close();
    }
} catch (IOException ex) {

```

Figura 15: Código para clasificar en función de la estabilidad

A continuación, por medio de otro experimento que hereda también de la clase *Experiment*, *EstudioFrentesNoDominados*, se obtienen los frentes de individuos no dominados para cada uno de los niveles de estabilidad obtenidos anteriormente. El experimento se encarga de llamar al algoritmo que se vaya a utilizar y éste, en lugar de evaluar los individuos, se encarga de leer los frentes de diferentes estabilidades, almacenando cada fila como un nuevo individuo, y obteniendo de todos ellos solamente los no dominados. Los frentes de Pareto de tres objetivos incluyen soluciones dominadas por otros puntos, que no son útiles y dificultan el análisis de los resultados, de ahí la necesidad de eliminarlas.

```

..
double [][] frenteF0 = metrica.readFront("EstabilidadSuperior."+ numero);

// Create the initial solutionSet
Solution newSolution;
for (int i = 0; i < frenteF0.length; i++) {
    newSolution = new Solution(2);

    for(int j= 0; j <2; j++){
        newSolution.setObjective(j, frenteF0[i][j]);
    }

    solutionSet.add(newSolution);
}

numero++;
Ranking ranking = new Ranking(solutionSet);
return ranking.getSubfront(0);

```

Figura 16: Código para la obtención del frente no dominado

De este modo, leyendo los ficheros para cada nivel de estabilidad, conseguíamos evaluar la robustez con distintos grados de distorsión. Pero, para la prueba final, donde verdaderamente se evalúa el funcionamiento de la robustez, era necesario aplicar otro experimento más: había que cambiar la función de fitness, de modo que evaluara solamente el rendimiento del mes que se está intentando predecir, incluyéndolo también para el cálculo del riesgo en la matriz de varianzas-covarianzas. A continuación, habría que evaluar si los individuos obtenidos de la primera prueba con dos objetivos se aproximan más que los obtenidos con los experimentos que incluyen la robustez. Sin embargo, tras la obtención de las soluciones no dominadas, sus individuos se perdían, por lo que era necesario recuperarlos para poder volver a evaluarlos con esta nueva función de fitness. Así pues, el primer paso consistía en identificar a qué línea pertenecía cada punto del frente de Pareto, y a cuál de los cuarenta y dos archivos. Una vez obtenido el fichero con esta información, por medio de otro experimento heredado de la interfaz *Experiment (EstudioEstabilidadEnDatoN)*, se leían y se escribían los ficheros de los individuos correspondientes, para cada uno de los tres niveles de estabilidad. Después se evaluaban con esta nueva función, y se obtenía el frente de Pareto con las soluciones no dominadas de dichos individuos. Llegados a este punto, los frentes de Pareto estaban preparados para ser representados gráficamente y comenzar su análisis.

```

int contador [] = leerNumeros("QueIndividuosAlto");
leerSalteado("VAR."+numero, contador);
Variable individuo [][] = readVariable("VARIABLEALTA."+numero);
double[][] frente =FUN."+numero);

// Create the initial solutionSet
Solution newSolution;
for (int i = 0; i < individuo.length; i++) {
    newSolution = new Solution(problem_, individuos[i]);
    problem_.evaluate(newSolution);
    evaluations++;
    solutionSet.add(newSolution);
} //for

numero++;
Ranking ranking = new Ranking(solutionSet);
return ranking.getSubfront(0);

} // execute

```

Figura 17: Código para obtener los individuos pertenecientes al frente no dominado

4.1.4.9. DISEÑO DE LOS OPERADORES GENÉTICOS

En este apartado se describen detalladamente todas las decisiones de diseño tomadas para la implementación de la aplicación, con fragmentos de código, esquemas y fórmulas que faciliten la comprensión de las explicaciones.

DISEÑO DEL CROMOSOMA:

El individuo está representado por una instancia de la clase *ArrayRealSolutionType*. Como su propio nombre indica, se trata de un array de valores reales, en el que cada gen de

la cadena cromosómica representa el porcentaje del dinero invertido para cada activo de la cartera. Los elementos con valor cero representan los activos no incluidos en ella. El número de ellos variará atendiendo al valor del parámetro de entrada correspondiente con la restricción de cardinalidad. El resto de los números de la cadena son valores reales comprendidos entre cero y uno (concretamente entre los umbrales de compra que se explicarán a continuación), representados con porcentajes en tantos por uno. La longitud de la cadena coincide con el del número de activos total que compone la cartera.

Como se ha adelantado, también debe cumplirse la otra restricción de Markowitz explicada en los objetivos ([1.2 Objetivos](#)). Para la inversión en cada activo debe tenerse en cuenta unos umbrales mínimos y máximos, cuyos valores se establecen en el fichero de entrada. Así, los pesos de las acciones de la cadena final, w_i , deben estar comprendidos dentro de estos umbrales, de tal forma que $w_i \in [l_i, l_s]$, donde l_i y l_s representan los límites de compra mínima y máxima, respectivamente ($l_i \geq 0.0$, $l_s \leq 1.0$ y $l_i \leq l_s$).

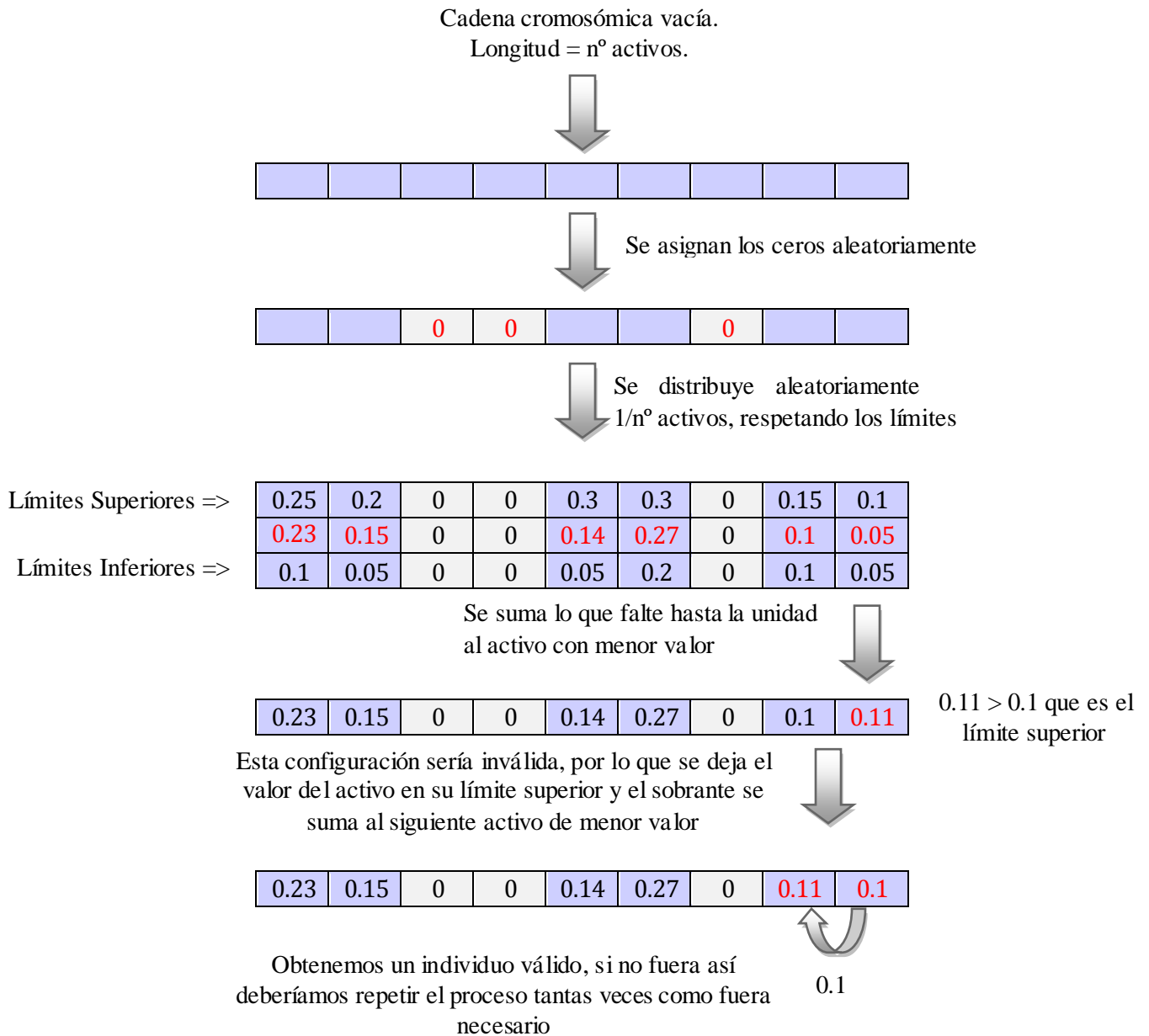
Y, finalmente, la última restricción que deben cumplir los pesos, es que la suma de todos ellos tiene que ser igual a 1.0, que representa el 100% del dinero invertido por el inversor, repartido entre las acciones incluidas en la cartera (aquéllas con valor distinto de cero): $\sum w_i = 1.0$, $w_i \geq l_i$ y $w_i \leq l_s$, $l_s \leq 1.0$ y $l_i \geq 0.0$, $\forall i$.

0.25	0	0	0.3	0.1	0	0.15	0	0.2
------	---	---	-----	-----	---	------	---	-----

Figura 18: Representación problema: diseño del cromosoma

El modo de implementación del individuo se resume con el siguiente pseudocódigo:

- 1) Los pesos con valor cero (que señalan aquellos activos que no se van a incluir dentro de la cartera en la población inicial), se establecen aleatoriamente, atendiendo al número máximo de activos leídos en el fichero de entrada.
- 2) Los pesos de los valores reales se intentan establecer de manera aleatoria y distribuida (de modo que inicialmente todos ellos se encuentran lo más uniformemente repartidos posible, con toda la aleatoriedad que permite tener que cumplir con toda las restricciones nombradas anteriormente). Cuando la suma de las acciones es menor que 1.0, el valor que falta hasta alcanzar dicha cantidad, se añade al peso con menor valor (distinto de 0, es decir, incluido en la cartera), siempre y cuando su nuevo valor no supere el umbral superior establecido para ese activo. Se descartan, como soluciones no factibles, aquellas cuya suma de sus instrumentos financieros superen el valor de la unidad, permitiéndose únicamente la conversión de soluciones no factibles en factibles cuando la suma no alcance la totalidad de la inversión (1.0), mediante un mecanismo de “reajuste”. La justificación de esta elección se ajusta a razones de simplicidad en los cálculos, a fin de conseguir la mayor aleatoriedad posible, pese a requerir una mayor complejidad.

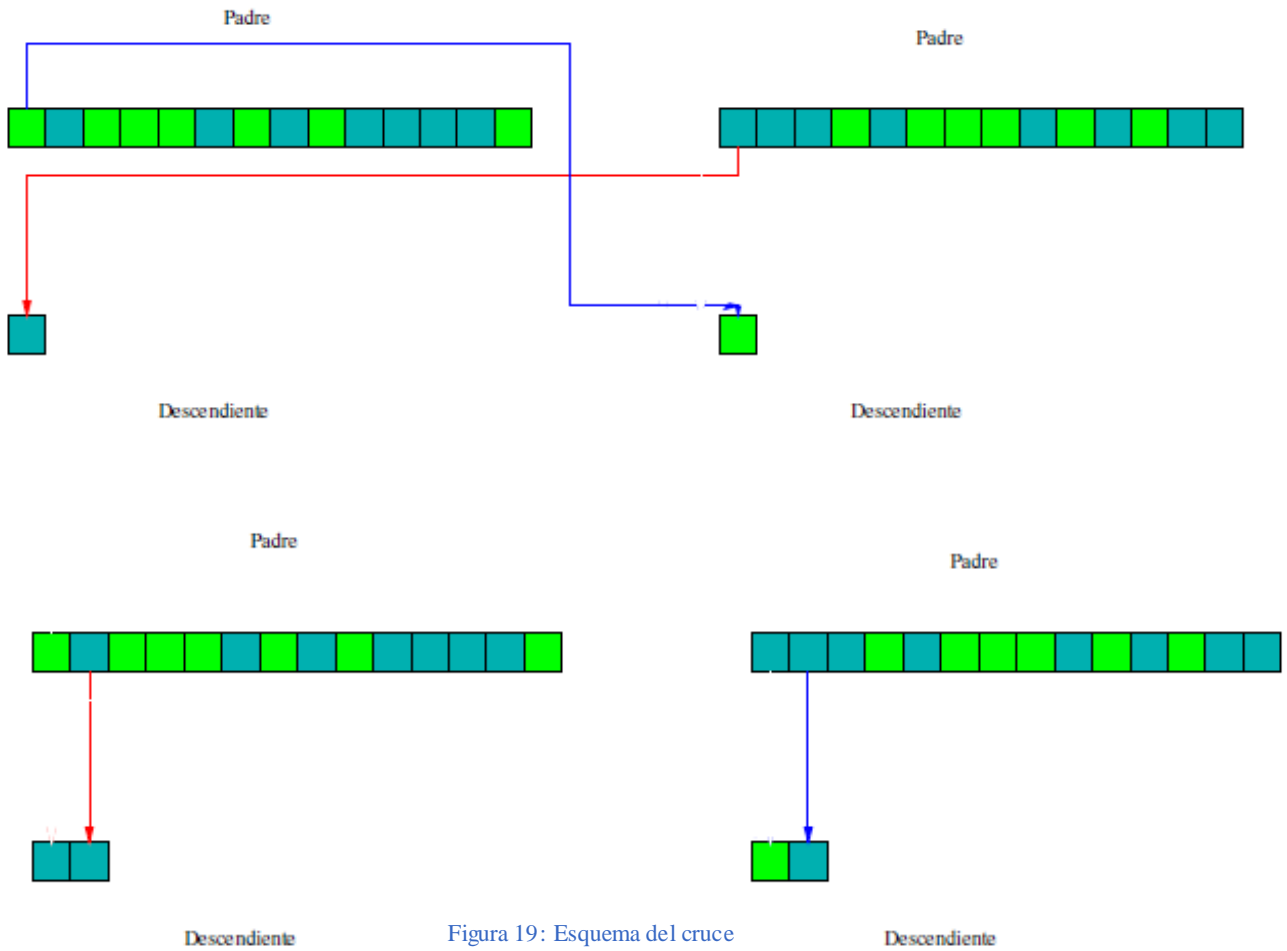


A la vista de la descripción de la implementación del cromosoma, no es de sorprender que muchos problemas de finanzas correspondan a la clase de los problemas NP-completos, debido a que tienen una estructura combinatoria equivalente (con respecto a reducciones en tiempo polinomial) a los bien conocidos problemas *NP-completos*. Por ejemplo, según *Schlottmann* y *Seese* [23], la selección de carteras con restricciones es equivalente al problema de la mochila o *Knapsack*, el cual se ha demostrado que es *NP-completo*.

Las operaciones de cruce y mutación se llevan a cabo en las clases *SBXCrossover* y *PolynomialMutation*, respectivamente. Pero, para poder evaluar la cadena resultante de ambos operadores genéticos, ésta debe constituir un individuo válido (para lo cual debe cumplir todas las restricciones explicadas anteriormente), de manera que puedan calcularse los valores objetivo correctamente, proporcionando individuos listos para competir por la supervivencia.

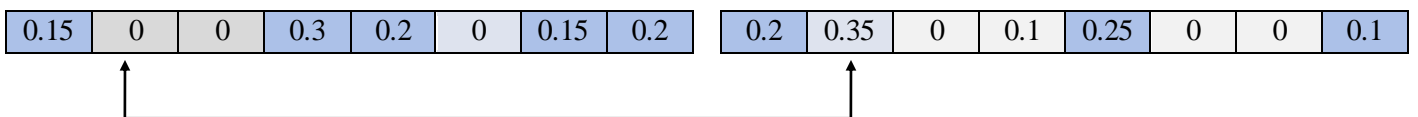
CRUCE

La operación de cruce llevada a cabo para los dos algoritmos genéticos empleados, es el cruce uniforme de cada dos posiciones de la cadena. El objetivo es tratar de conseguir individuos con el 50% de las características de cada uno de los progenitores.

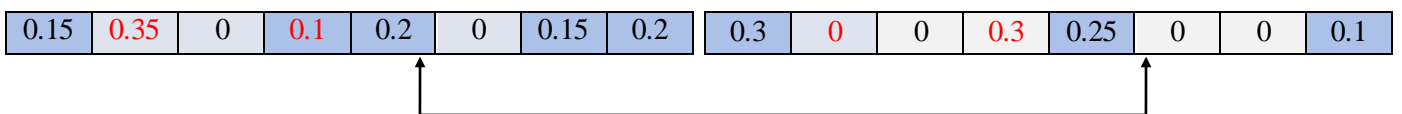


A continuación se representa este esquema con los valores de la cadena cromosómica para una cartera de inversión:

Se cruzan los elementos pares de cada individuo



Este intercambio es aleatorio, tiene un 50% de probabilidades de realizarse

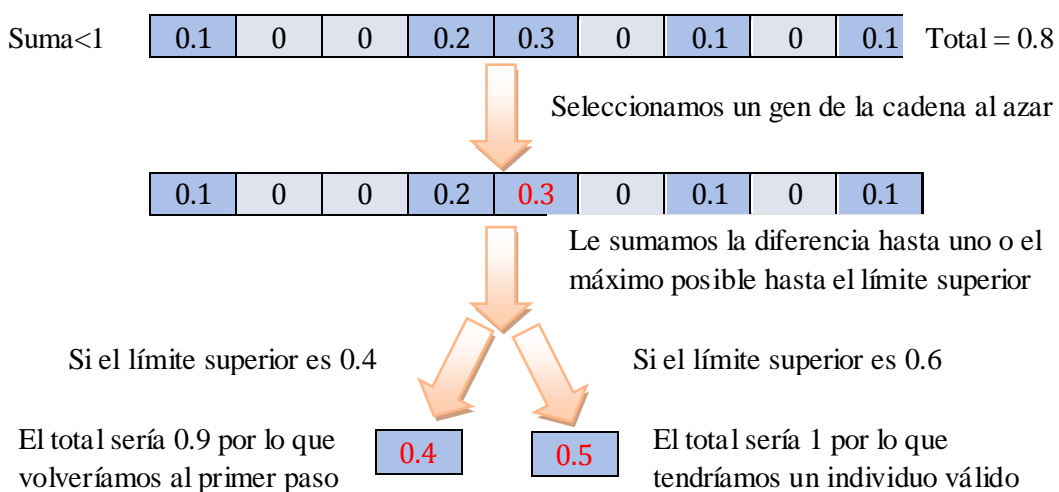


El proceso continuaría hasta llegar al final de los individuos

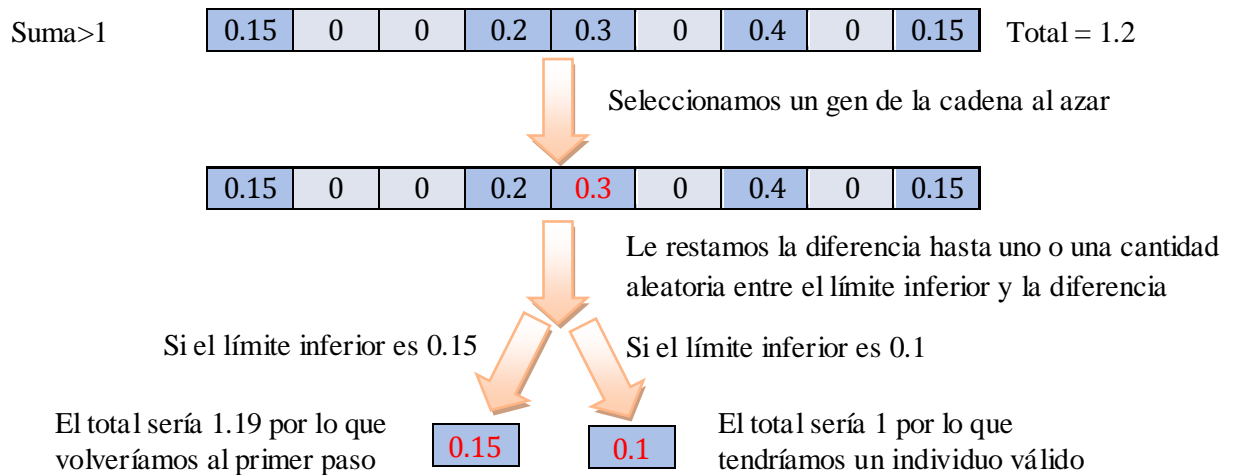
Una vez realizado el cruce de las cadenas cromosómicas de los individuos padre, se generan dos nuevos individuos. Su aplicación de forma selectiva sobre los progenitores, permite que prevalezcan sus características en su descendencia, pero mezcladas con las características de otros buenos padres. Sin embargo, tras su aplicación, se obtienen dos cadenas con pesos comprendidos entre su umbral máximo y mínimo, pero cuya suma es prácticamente imposible que coincida con la unidad. Además, puede incumplirse también la restricción de cardinalidad, al obtenerse un individuo que invierte en más activos de los permitidos (con menor número de ceros de los que debería tener). Por este motivo, es necesario efectuar una serie de rectificaciones o reajustes, mediante la aplicación de un algoritmo de reparación y manipulación de restricciones, para la obtención de individuos descendientes válidos.

El algoritmo de reparación se encarga, en primer lugar, de conseguir que todos los pesos del individuo sumen 1.0. Se restará siempre a la suma de sus pesos la cantidad necesaria hasta alcanzar la unidad, de modo que se irá actualizando esta cifra (llamada *sobrante* en el código) hasta conseguir que su valor sea cero. Para ello, se elegirá una posición aleatoria de la cadena, con valor distinto de cero, si el sobrante es positivo, o cualquier valor en caso contrario, y se le restará la cantidad que dista hasta la unidad (el *sobrante*). Si en el resultado obtenido el peso queda fuera de sus umbrales por sobrepasar el máximo, se le asignará el valor de este umbral o si resulta ser inferior al mínimo, se le asignará un valor próximo a él, pero algo superior, a fin de cumplir con la restricción. El motivo de no asignar el umbral mínimo exacto, es proporcionar mayor variabilidad genética a la cadena. A continuación, si tras el último reajuste el *sobrante* es positivo, se le restará la diferencia entre el peso original para esa posición, y su límite inferior o, si es negativo, se le sumará la diferencia entre el umbral superior de dicha posición, y el peso que tenía originalmente. Este proceso se repite hasta que la cantidad obtenida tras la resta del *sobrante* al peso original quede dentro de los límites de compra máxima y mínima, tras lo cual sobrante será cero. Entonces se habrá cumplido el objetivo: la suma de todos los valores del individuo es igual a uno, con todos ellos dentro de los límites correspondientes. Con motivos de facilitar la comprensión de esta descripción, se muestra el un esquema de ejemplo y el código del algoritmo al que se hace alusión:

Individuo inválido, ya que la suma de activos menor que uno:



Individuo inválido, ya que la suma de activos mayor que uno:



A continuación se muestra un ejemplo del código asociado a este esquema:

```
public XReal rectificar (double suma, XReal individuo) throws JMException{
    double sobrante = suma - 1;
    while(sobrante > 0){
        //cogemos un activo a rectificar aleatorio
        int cartera = ((int) (Math.random()*individuo.getNumberOfDecisionVariables()));
        if (individuo.getValue(cartera) != 0.0){
            if(individuo.getValue(cartera) - sobrante < individuo.getLowerBound(cartera)){
                double diferencia = (individuo.getUpperBound(cartera) -
                    individuo.getLowerBound(cartera))/3;
                double limiteInferior = individuo.getLowerBound(cartera);
                double limiteSuperior = limiteInferior + diferencia;
                double valorMenor=(Math.random()*(limiteSuperior-limiteInferior)+limiteInferior);
                sobrante -= (individuo.getValue(cartera)-individuo.getLowerBound(cartera));
                individuo.setValue(cartera, individuo.getLowerBound(cartera));
            }else {
                if(((individuo.getValue(cartera)-sobrante) >= individuo.getLowerBound(cartera))
                    &&(individuo.getValue(cartera)-sobrante) <= individuo.getUpperBound(cartera)){
                    individuo.setValue(cartera, individuo.getValue(cartera)-sobrante);
                    sobrante=0;
                }
            }
        }
    }
    while(sobrante < 0){
        //cogemos un activo a rectificar aleatorio
        int cartera = ((int) (Math.random()*individuo.getNumberOfDecisionVariables()));
        if((individuo.getValue(cartera) - sobrante) > individuo.getUpperBound(cartera)){
            sobrante += (individuo.getUpperBound(cartera) - individuo.getValue(cartera));
            individuo.setValue(cartera, individuo.getUpperBound(cartera));
        }else {
            if(((individuo.getValue(cartera)-sobrante) >=individuo.getLowerBound(cartera))
                &&(individuo.getValue(cartera)-sobrante)<=individuo.getUpperBound(cartera)){
                individuo.setValue(cartera, individuo.getValue(cartera) - sobrante);
                sobrante=0;
            }
        }
    }
}
```

Figura 20: Código para rectificar individuos que no sumen uno

Pero, tras este algoritmo, todavía quedaría pendiente por asegurar una restricción: la de cardinalidad. De modo que habría que comprobar el número máximo de activos en los que se invierte, añadiéndose tantos ceros como sean necesarios si el número de ellos es menor de lo que correspondería (se invierte en más activos del máximo permitido). Se convertirán en cero los menores pesos de la cadena, hasta alcanzar el número estipulado como parámetro en el fichero de entrada. A continuación deberán realizarse nuevamente los reajustes necesarios para que la suma de todos los pesos sea igual a uno (sin incluir aquellos con valor cero), y respetando los umbrales de compra máxima y mínima para cada uno de ellos. Se calcula cuanto difiere la suma de los pesos de la cadena con respecto a la unidad. Si la diferencia entre el valor del peso a rectificar y el límite superior es inferior a la distancia con respecto a la unidad, se le suma este exceso. En caso contrario, si supera el valor 1.0, se le suma la cifra que dista hasta alcanzarlo. De este modo, se va reduciendo esta diferencia, denominada margen, hasta que su valor llegue a ser cero, ya que entonces la suma de los pesos de la cadena habrá alcanzado la unidad. Su implementación quedaría como se muestra a continuación:

El individuo tiene un sólo cero, cuando el mínimo de ceros es 3

0.13	0.1	0	0.15	0.11	0.14	0.16	0.1	0.11
------	-----	---	------	------	------	------	-----	------

Se ponen a cero los activos con menor valor hasta llegar al número mínimo de ceros

0.13	0	0	0.15	0.11	0.14	0.16	0	0.11
------	---	---	------	------	------	------	---	------

Se divide lo que falta hasta la unidad (0.2) entre los activos diferentes de cero.

0.16	0	0	0.18	0.14	0.17	0.19	0	0.14
------	---	---	------	------	------	------	---	------

Si al sumar se sobrepasa el límite superior se le daría al activo el valor del límite superior y se repetiría el proceso de repartir lo que falta entre todos los activos

Suponemos que el límite es 0.16

0.16	0	0	0.18	0.14	0.17	0.16	0	0.14
------	---	---	------	------	------	------	---	------

Se divide lo que falta hasta la unidad (0.05) entre el resto de activos distintos de cero

0.17	0	0	0.19	0.15	0.18	0.16	0	0.15
------	---	---	------	------	------	------	---	------

Finalmente se obtiene un individuo válido

A continuación se muestra el código correspondiente a este esquema:

```

//nos aseguramos de que el individuo tenga el número de 0s correctos (para que sea válido)

contadorCeros = 0;
menorValor = 1.0;
int contadorDividendos = 0;
boolean rectificado = false;

// contamos cuantos ceros hay en la cadena
for(int t = 0; t< numberOfVariables; t++){
    if(offsl1.getValue(t) == 0){
        contadorCeros++;
    }
}

/* si hay menos ceros de los que debería se busca la posición con menor valor (¡=0)
hasta alcanzar el número de ceros especificado en el parámetro del fichero.
*/
while(contadorCeros < ceros){
    rectificado = true;
    for(int t = 0; t< numberOfVariables; t++){
        if(menorValor > offsl1.getValue(t) && offsl1.getValue(t) != 0.0){
            menorValor = offsl1.getValue(t);
            posicion = t;
        }
    }
    offsl1.setValue(posicion, 0.0);
    contadorCeros++;
}

/* Con el nº de 0s correcto, se hacen las rectificaciones correspondientes para que
el total de los pesos de la cadena sume 1.0, respetándose el resto de las restricciones.
*/
double exceso = 0.0;
double total = 0.0;
for(int t = 0; t< numberOfVariables; t++){
    if(offsl1.getValue(t) != 0){
        contadorDividendos++;
        total += offsl1.getValue(t);
    }
}

//primero se calcula cuanto difiere la suma de los pesos con respecto a la unidad
while(exceso > 0 && rectificado){
    double margen = exceso / contadorDividendos;
    for(int i = 0; i < numberOfVariables; i++){
        if((offsl1.getValue(i) + margen) <= offsl1.getUpperBound(i) && offsl1.getValue(i) != 0.0){
            offsl1.setValue(i, offsl1.getValue(i) + margen);
            //De este modo se va reduciendo el margen hasta que se llega a cero.
            exceso -= margen;
        } else if((offsl1.getValue(i) + margen) > offsl1.getUpperBound(i) &&
offsl1.getValue(i) != 0.0) {
            exceso -= (offsl1.getUpperBound(i) - offsl1.getValue(i));
            offsl1.setValue(i, offsl1.getUpperBound(i));
        }
    }

    total = 0.0;
    contadorDividendos = 0;
    for(int t = 0; t< numberOfVariables; t++){
        if(offsl1.getValue(t) != 0){
            contadorDividendos++;
            total += offsl1.getValue(t);
        }
    }
    exceso = 1.0 - total;
}
}

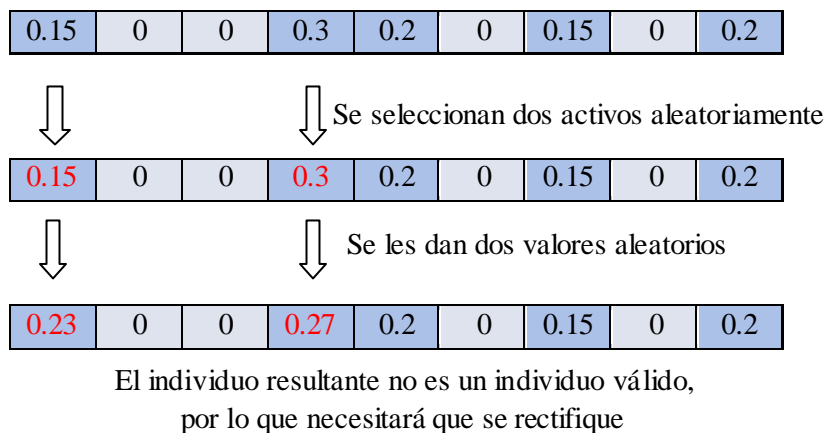
```

Figura 21: Código para rectificar individuos que no cumplan la restricción de cardinalidad

MUTACIÓN

Es necesaria la existencia de otro operador para lograr introducir variabilidad genética a los individuos obtenidos tras el cruce, y evitar la convergencia prematura. La mutación consiste en la variación de uno o más alelos del gen. Su aplicación de forma aleatoria a diferentes puntos de la cadena cromosómica, produce individuos con pequeñas variaciones con respecto al individuo original. Para este problema, la mutación implica el cambio de dos alelos del gen, pues la modificación de un peso, implica necesariamente la de otro para ajustar la suma de todos ellos a uno. Por tanto, en primer lugar se seleccionan aleatoriamente los dos alelos que mutarán, cambiando su valor por otro comprendido entre los umbrales de compra correspondientes a la restricción de su posición. Y, a continuación, deben realizarse reajustes análogos a los que se llevaron a cabo tras el operador de cruce, a fin de conseguir un individuo válido para evaluar. Así pues, deben respetarse todas las restricciones, incluyendo que la suma de todos sus pesos sume 1.0. De este modo, al igual que ocurría con el operador de cruce, se distinguirán tres partes en su implementación:

- 1) La mutación propiamente dicha, donde cada peso se encuentra comprendido dentro de los umbrales correspondientes.
- 2) Los reajustes que aseguren que la suma de todos ellos sume 1.0.
- 3) Y, finalmente, el reajuste necesario para que se cumpla la restricción de cardinalidad.



A continuación se muestra el código asociado al esquema. El código de los dos últimos puntos no se incluye por ser análogo al de las rectificaciones del cruce.

```

// muta dos carteras
posicion1 = (int) (Math.random() * solution.numberOfVariables());
posicion2 = (int) (Math.random() * solution.numberOfVariables());

while(posicion1 == posicion2) {

    posicion2 = (int) (Math.random() * solution.numberOfVariables());
    posicion1 = (int) (Math.random() * solution.numberOfVariables());
}

suma-=x.getValue(posicion1);
suma-=x.getValue(posicion2);

x.setValue(posicion1, Math.random());
x.setValue(posicion2, Math.random());
suma+=x.getValue(posicion1);
suma+=x.getValue(posicion2);

while (x.getValue(posicion1)<x.getLowerBound(posicion1) || x.getValue(posicion1)>
x.getUpperBound(posicion1)|| x.getValue(posicion2) <
x.getLowerBound(posicion2) || x.getValue(posicion2) >
x.getUpperBound(posicion2)){

    suma-=x.getValue(posicion1);
    suma-=x.getValue(posicion2);
    x.setValue(posicion1, Math.random());
    x.setValue(posicion2, Math.random());
    suma+= x.getValue(posicion1);
    suma+= x.getValue(posicion2);
}

/* Rectificaciones */

```

Figura 22: Código de la mutación

4.1.4.10. FUNCIONES OBJETIVO:

Como se ha ido repitiendo a lo largo del documento, son tres los objetivos que se pretenden implementar para la optimización de la cartera de inversiones. A continuación se describirá cada uno de ellos de manera detallada, con fragmentos de código y explicaciones adicionales que permitan la comprensión de ciertos aspectos técnicos del dominio, justificando el modo en que se ha llevado a cabo su implementación. Los objetivos se describen en el mismo orden de aparición que en el fichero de salida (FUN) y, por consiguiente, que en los ejes de coordenadas que se mostrarán en el capítulo de la experimentación.

FUNCIÓN DE FITNESS:

Es la encargada de llamar a las tres funciones asociadas con la implementación de cada uno de los objetivos a resolver. Por defecto, el framework minimiza las funciones objetivo. Por tanto, para la implementación del rendimiento, el único objetivo a maximizar, es necesario hacer la siguiente conversión: $Max(f(x)) = -Min(f(-x))$. El código de esta función quedaría:


```

public void evaluate(Solution solution) throws JMException, MatrixIndexException {
    XReal variable = new XReal (solution);
    try{
        double [] f = new double[numberOfObjectives_]; //variable con valor 3.

        double h = this.evalRiesgo(variable); //minimizamos, por defecto lo que hace JMetal
        double g = this.evalRendimiento(variable); //maximizamos: Max(f(x))=- Min(f(x));
        double j = this.evalRobustez(variable, solution);

        f[0] = h;
        f[1] = -g; //maximizamos
        f[2] = j;

        solution.setObjective(0, f[0]);
        solution.setObjective(1, -1* f[1]); //estamos maximizando.
        solution.setObjective(2, f[2]);
    } catch (IOException e){
        e.printStackTrace();
    }
}

```

Figura 23: Código para la asignación de objetivos

A) FUNCIÓN OBJETIVO RIESGO:

De forma básica, el algoritmo trata de implementar la fórmula del riesgo asociado a la inversión de una cartera, descrito formalmente en el apartado 2.1 (Contexto del problema), con el objetivo de minimizarlo.

$$\sigma_p^2 = \sum_i w_i^2 \sigma_i^2 + \sum_i \sum_{j \neq i} w_i w_j \sigma_i \sigma_j \rho_{ij},$$

Cuantitativamente, el riesgo se representa con la covarianza, o con la desviación estándar (raíz cuadrada de la varianza):

$$\sigma_p = \sqrt{\sigma_p^2}$$

Como se puede ver, la desviación estándar de una cartera depende no sólo de las varianzas de los instrumentos financieros individuales, sino también de la covarianza de los rendimientos posibles entre cada par de instrumentos financieros.

Para implementar la matriz de covarianzas y el coeficiente de correlación, antes se ha tenido que leer (en la clase *LecturaDatosEntrada*) los datos con las series de las rentabilidades. Estos datos serán las columnas de la matriz sobre la que se calculará la matriz de covarianza.

```
RealMatrix matriz = new RealMatrixImpl(numeroMeses, (int)numeroActivos);

for (int i = 0; i < numeroMeses; i++){
    for(int j = 0; j < numeroActivos; j++){
        matriz.setEntry(i, j, rendimiento[i][j]);
    }
}
```

Figura 24: Creando la matriz de covarianzas

La covarianza indica en qué medida varía una acción respecto a la otra. De esta forma, si la covarianza es positiva, quiere decir que cuando una acción sube, la otra también tiende a subir; si la covarianza es negativa, quiere decir que cuando “a” sube “b” tiende a bajar. Si la covarianza es próxima a cero, quiere decir que las dos acciones no están relacionadas.

La implementación de la matriz de covarianzas:

$$Cov_{ab} = \frac{1}{n-1} \cdot \sum_{i=1}^n (R_{ai} - E[R_a]) \cdot (R_{bi} - E[R_b])$$

y del factor de correlación ($\rho_{ab} = \frac{Cov_{ab}}{\sigma_a \cdot \sigma_b}$) es trivial gracias al empleo de la librería Commons-Math, invocada desde java con la siguiente llamada:

```
import org.apache.commons.math.stat.correlation.*;
```

El código quedaría tan simplificado como sigue:

```
Covariance covarianza = new Covariance(matriz);
resultadoCovarianza = new RealMatrixImpl((int)numeroActivos, (int)numeroActivos);
resultadoCorrelacion = new RealMatrixImpl((int)numeroActivos, (int)numeroActivos);
resultadoCovarianza = covarianza.getCovarianceMatrix();
PearsonsCorrelation indice = new pearsonsCorrelation(resultadoCovarianza);
resultadoCorrelacion = indiceCorrelacion.computeCorrelationMatrix(matriz);
```

Figura 25: Usando la librería Commons-Math para crear la matriz de covarianzas

Donde la matriz de origen, pasada como parámetro a Covariance, es la matriz definida en la figura 24, cuyas columnas se obtienen a partir de las series de rentabilidades leídas del fichero de entrada.

Una vez definidos los parámetros necesarios para resolver la fórmula del riesgo, mostramos la implementación llevada a cabo para su resolución:

```

public double evalRiesgo(XReal solution) throws MatrixIndexException, JMException{
    double riesgo = 0.0;
    double [] vectorDesviaciones = new double [numberOfVariables_];
    for (int i=0 ; i<numberOfVariables_ ; i++){
        for(int j=0; j<numberOfVariables_ ; j++){
            if(i == j){
                vectorDesviaciones [i] = Math.sqrt(datos.getMatrizCovarianza().getEntry(i,j));
            }
        }
    }

    for (int i=0 ; i<numberOfVariables_ ; i++){
        for(int j=0; j<numberOfVariables_ ; j++){
            if(i == j){
                riesgo += datos.getMatrizCovarianza().getEntry(i, i)*
                    * Math.pow(solution.getValue(j),2);

            }else {
                riesgo += vectorDesviaciones [i]*vectorDesviaciones [j]* solution.getValue(i) *
                    * solution.getValue(j)* datos.getIndiceCorrelacion().getEntry(i, j);
            }
        }
    }

    return riesgo;
}

```

Figura 26: Código para la evaluación del riesgo

B) FUNCIÓN RENDIMIENTO:

En esta segunda función objetivo, la única a maximizar, se implementa el rendimiento previsto para el conjunto de las series de rendimientos mensuales que componen los ocho activos de la cartera, situados en el fichero de entrada.

De este modo, el rendimiento previsto de cada índice del fichero de entrada, es la media de los rendimientos mensuales que tuvo el índice durante todos los años que comprenden las muestras. Por tanto, la rentabilidad de una cartera será igual a la suma ponderada de las rentabilidades de los activos que la componen. Se ponderarán las rentabilidades por el peso específico que cada activo tiene en la cartera, tal y como se expresa en la siguiente ecuación, formalizada en el punto [2.1](#) Contexto del Problema:

$$E(Rp) = \sum_{i=1}^n w_i \cdot E(R_i) = w_1 \cdot E(R_1) + w_2 \cdot E(R_2) + \dots + w_n \cdot E(R_n)$$

A continuación se muestra el código que implementa esta función:

```
public double evalRendimiento(XReal solution) throws JMException, FileNotFoundException{
    double [][] rendimiento = datos.getRendimiento();
    double [] suma_rendimiento = new double [numberOfVariables_];
    double suma = 0.0;
    double rendimientoTotal = 0.0;
    for (int j = 0; j<numberOfVariables_; j++) {
        for (int i = 0; i<datos.getnumeroMeses(); i++) {
            suma += rendimiento [i][j];
        }

        suma_rendimiento [j] = suma/datos.getnumeroMeses();
        rendimientoTotal+= suma_rendimiento [j]*solution.getValue(j);

        suma = 0.0;
    }

    return rendimientoTotal;
}
```

Figura 27: Código para la evaluación del rendimiento

C) FUNCIÓN ROBUSTEZ:

La visión tradicional es asumir que las características de riesgo históricas persistirán en un futuro. Sin embargo, cualquier cambio producido en el mercado altera el resultado que contra todo pronóstico se preveía para ese activo, atendiendo a su evolución histórica. Los perfiles de beneficio-riesgo varían significativamente ante cualquier cambio inesperado que se produzca en el mercado.

Nuestra intención es utilizar la robustez para **minimizar** la distancia que podría existir entre el par rentabilidad-riesgo de los parámetros obtenidos de los datos históricos, y el que se obtendría al producirse alguna variación imprevista.

Para calcular estas variaciones, se introducen en el sistema dos parámetros de distorsión:

- El **radio** o rango de variación entre las carteras que se están evaluando y las resultantes al introducir ruido.
- El **exponente** al que se elevan estas distorsiones, para penalizar, en mayor medida, las que tengan un valor más alto.

A continuación se calculará la distancia de **Mahalanobis** entre la cartera original y cada una de las resultantes al introducir ruido sobre ella. La métrica es el promedio de todas estas distancias. Se considera que el sistema se está evaluando bajo una situación de normalidad en el mercado, por lo que se espera que las variaciones en él sean mínimas. Por este motivo, se minimizan los resultados obtenidos.

A continuación, se explican los pasos que se han seguido para implementar esta última función objetivo que pretende proporcionar estabilidad al problema de optimización de carteras:

- 1) En primer lugar, en la clase SPEA2 se crea un conjunto de individuos aleatorios para calcular la desviación estándar en el espacio de objetivos (rentabilidad-riesgo). Como nuestra intención es introducir cierta variabilidad que simule cualquier cambio que pueda producirse en el mercado, se calculará la matriz de covarianzas de la fórmula de Mahalanobis con otro conjunto de individuos distinto al que se está evaluando.
- 2) El resto de la fórmula, se implementa en la clase *CarteraInversion*, en el cuerpo de la tercera función objetivo:
 - a) Se dispone de la cartera actual a evaluar.
 - b) Se distorsiona la cartera a evaluar introduciendo ruido. Para ello, se crea un conjunto de individuos similares al que se está evaluando, variando los porcentajes de cada activo alrededor de un radio $[-r, r]$ en torno a ellos. El valor de radio viene determinado en el fichero de entrada.
 - c) Calculo la distancia de Mahalanobis entre a y b (el individuo real y el distorsionado).
 - d) Se eleva a n cada distancia resultante en c. El valor de la potencia viene dado en el fichero de entrada.
 - e) Se repiten los pasos b, c y d para todas las distorsiones que se han generado sobre el individuo a evaluar.
 - f) Se calcula la distancia promedio para todas las distancias de Mahalanobis obtenidas en e.

4.1.4.11. REMUESTREO DE PARÁMETROS:

El remuestreo hace referencia al modo en que se seleccionan los datos de entrada para el cálculo de las funciones objetivo rentabilidad-riesgo. De una forma muy simple, la simulación consiste en repetir un proceso de generación de muestras, con reemplazamiento, un número suficientemente elevado de veces, seleccionando de manera aleatoria las series de rendimientos mensuales. Este proceso se repite hasta completar el número de series existentes en el fichero de entrada, para poder realizar inferencias. En definitiva, mediante la repetición y generación de muestras de datos se crea un conjunto de “escenarios posibles”, para tratar de evitar hiperespecializar la solución en uno solo, que podría ajustarse o no a la realidad. El problema de la hiperespecialización para un único escenario es que el resultado podría ser potencialmente muy sensible a pequeñas desviaciones entre el escenario previsto y el que se da en la realidad, por lo que la solución obtenida sería poco robusta.

4.1.4.12. FICHERO DE ENTRADA:

Es el fichero contiene todos los parámetros configurables del programa, con los que se jugará a fin de intentar conseguir los mejores resultados.

En primer lugar, se encuentra el modo de ejecución del programa: mediante remuestreo o sin él.

A continuación se encuentran los parámetros que hacen referencia a las restricciones de cardinalidad y de los umbrales máximos y mínimos de compra.

Posteriormente, se lee el número de períodos de tiempo que abarcan las series de rendimientos mensuales, y el número de activos que componen la cartera. Y, a continuación, se leen los parámetros relacionados con el objetivo de la estabilidad: el número de individuos que rodearán al que se está evaluando, el radio en torno al cual se situarán y el exponente al que se elevará la distancia final obtenida entre estos individuos y aquel cuya estabilidad se quiere valorar. Finalmente, se encuentran las series de datos que componen los activos a optimizar.

Capítulo 5

Experimentación

5. EXPERIMENTACIÓN

En este capítulo se presenta el trabajo experimental realizado. Para ello, se describen los parámetros de distorsión utilizados para introducir pequeños cambios a la composición de la cartera, y se presentan los resultados obtenidos al experimentar con ellos.

En cada caso de prueba, se aplica un parámetro de distorsión diferente, se explica qué características lo hacen interesante en este contexto y cuáles son las dispersiones de la rentabilidad y riesgo obtenidas para cada uno de ellos, con respecto a la cartera original.

Se presentan tablas donde se recogen medidas estadísticas de estas dispersiones, agrupadas en tres niveles de estabilidad, en función de las alteraciones que produzcan en los resultados. A continuación, se realizará una comparativa para los distintos valores asignados a los parámetros de distorsión causantes de las dispersiones.

Se incluyen representaciones gráficas para algunos frentes de Pareto analizados en estas tablas, con el objetivo de ilustrar las alteraciones de las soluciones obtenidas con cada parámetro, ante los errores de estimaciones cometidos.

Finalmente, se presentan métricas que permiten comparar la calidad de los frentes de Pareto obtenidos.

Al término de cada uno de los casos de prueba definidos, se recogen las conclusiones obtenidas de los resultados experimentales.

5.1. MUESTRA UTILIZADA

En este apartado presentamos la muestra a utilizar en la parte experimental del trabajo. La muestra se compone de ocho series de rendimientos mensuales, de ocho índices financieros durante el período comprendido entre el mes de Diciembre de 1979 y el mes de Marzo de 2007.

Estos datos representan distintas familias de activos muy conocidos en los que se pueden invertir. Los primeros cuatro ("*Frank Russell 2000 Growth*", "*Frank Russell 2000 Value*", "*Frank Russell 1000 Value*", "*Frank Russell 2000 Growth*"), son índices de acciones estadounidenses y representan empresas cotizadas de tamaño pequeño (2000) y grande (1000) que se caracterizan por tener un gran potencial de crecimiento (*growth*) o por tener cotizaciones que están por debajo de lo que se presume que pueden valer (*value*). El quinto, mantenido por Goldman Sachs ("*GSCI*"), es un índice del mercado de materias primas. El siguiente, "*MSCI EAFE*", es un índice de Morgan Stanley que representa las acciones en mercados internacionales (no estadounidenses). Y los dos últimos ("*US Corp*" y "*US TRSY /AGCY*"), son índices de BofA Merrill Lynch que representan dos tipos de instrumentos de renta fija estadounidense. El primero, cubre instrumentos de deuda de empresas y el último de deuda pública. Estos datos se han obtenido de la base de datos DATASTREAM de la biblioteca de la universidad.

5.2. PROCESO DE EXPERIMENTACIÓN

Antes de comenzar con el proceso de experimentación propiamente dicho, se llevaron a cabo un conjunto de pruebas que verificaran el buen funcionamiento del sistema desarrollado. Como actualmente no existe ninguna aplicación enfocada en el objeto de estudio de este proyecto, no se puede hacer un contraste directo con ningún otro sistema para validar los resultados. Por lo tanto, nos hemos visto obligados a evaluar el funcionamiento de la aplicación descomponiéndola en módulos.

La primera prueba se hizo relajando las condiciones del problema, de modo que tratara de resolver el modelo de Markowitz eliminando las restricciones. De este modo, pudieron validarse los resultados de las dos funciones objetivos del modelo. Se obtenía un frente prácticamente idéntico al conseguido por la aplicación JPortfolio3. Se trata de un programa basado en optimización multiobjetivo, mediante algoritmos genéticos, que obtiene frentes de Pareto Óptimos maximizando el rendimiento y minimizando el riesgo. Se emplearon más aplicaciones similares a ésta para tal fin (como, por ejemplo, Portfolio_Optimization.xls). Una vez validados los resultados de estas dos funciones objetivo, se hizo uso de un código en Matlab, para comprobar la salida del módulo que calcula la distancia de Mahalanobis. La implementación relativa a los operadores genéticos (la creación de los individuos de la población inicial, el cruce y la mutación), así como las restricciones de impuestas al modelo de Markowitz, se pudieron validar mostrando trazas de la ejecución del algoritmo.

Una vez verificado el correcto funcionamiento de todos los módulos del sistema, y tras comprobar que los frentes de Pareto obtenidos mostraban resultados coherentes, se dio por validado el sistema y pudo plantearse el plan de pruebas de la fase de experimentación.

5.2.1 PLAN DE EXPERIMENTACIÓN

La aplicación se ha evaluado en función de un conjunto de casos de prueba mediante los cuales se determinará si el sistema es tolerante a pequeñas equivocaciones en los parámetros del modelo de Markowitz. A continuación se definen los experimentos y casos de pruebas que se han llevado a cabo:

Inicialmente, el objetivo es demostrar que lo que se ha leído en la literatura sobre SPEA2 [22], [26] es cierto y que, realmente, se adapta mejor que NSGA-II a los problemas de optimización de carteras para este fin se consideraron, únicamente, las dos primeras funciones objetivos descritas en el apartado 4.1.4.10 (Funciones objetivo), para maximizar el rendimiento y minimizar el riesgo. Además, se utilizan dos restricciones: la restricción de cardinalidad, de modo que se invierte en seis de los ocho activos disponibles, y la restricción de umbrales de compra, con un máximo del 60% y un mínimo del 5% en todos los activos. Se trabajará con los algoritmos presentados con anterioridad, sobre n-1 datos del total de las series de rendimientos.

El enfoque general adoptado en los ensayos de cada uno de los algoritmos genéticos, consiste en llevar a cabo 42 ejecuciones independientes con cada uno de los dos algoritmos explicados. Se trabajará, en un principio, con dos objetivos (maximizando el rendimiento y minimizando el riesgo), hasta que se alcance el criterio de parada (un determinado número de generaciones). A continuación, se establece una media, mediana y desviación estándar

³ Puede descargarse en <http://www.mhsatman.com/jportfolio/>

para cada uno de los algoritmos, en términos de Spread e Hipervolumen. Los resultados se registrarán para 3000 evaluaciones, con un tamaño de población de 100 individuos.

Puesto que se pretende centrar la atención en el estudio del tercer objetivo (robustez), se repetirá el proceso anterior con el algoritmo SPEA2 y los tres objetivos. Se utilizará un radio del 5%, como rango de variación para las distorsiones generadas en la cartera evaluada. El número de distorsiones para cada individuo a evaluar ascenderán a 25. Se calcularán las mismas métricas que en el experimento con dos objetivos, y se evalúan dichas métricas (Spread e Hipervolumen), teniendo en cuenta solamente la rentabilidad y el riesgo.

Para facilitar el análisis de la estabilidad, en función de los distintos parámetros de distorsión introducidos (radio, exponente y remuestreo), los individuos de los frentes de tres objetivos se agruparán en tres niveles de estabilidad diferentes (alta, media y baja). Cuanto mayor sea la estabilidad, mayor robustez se espera en las soluciones alcanzadas. Una dispersión promedio más baja en las soluciones finales, supondrá menos alteraciones entre los resultados estimados y los alcanzados y, por lo tanto, una mayor estabilidad en el sistema.

5.2.1.1 EXPERIMENTO BASE

1. **Caso 1:** la configuración anterior servirá como caso de estabilidad básica para el resto de las pruebas, en torno al cual irán dirigidas las comparaciones de los resultados obtenidos con los demás experimentos. Los parámetros de esta configuración serán fijos en la mayor parte de los casos de prueba.

5.2.1.2 SENSIBILIDAD AL RADIO

2. **Caso 2:** ídem que el caso 1, pero duplicando el rango de variación del conjunto de las 25 carteras creadas a similitud de la que se está evaluando.
3. **Caso 3:** ídem que el caso 1, pero triplicando dicho rango de variación.

El propósito de este caso de prueba es analizar cómo influyen las distorsiones realizadas sobre las carteras originales, suponiendo mayores errores en la estimación de los parámetros. Cuanto mayor sea la distorsión introducida, mayor será el margen de equivocación tolerado por el sistema y, por tanto, mayor robustez se le estará exigiendo.

5.2.1.3 SENSIBILIDAD AL EXPONENTE

4. **Caso 4:** ídem que el caso 1, pero elevando el exponente al cuadrado.
5. **Caso 5:** ídem que el caso 1, pero elevando el exponente al cubo.

El objetivo de este caso de prueba es analizar cómo influye, en la minimización del objetivo, elevar a un exponente las distancias de Mahalanobis obtenidas entre la cartera original y las distorsionadas en torno a ella. Este parámetro se toma como medida de penalización para distancias grandes. Cuanto mayor sea el exponente, mayor será la penalización y, por tanto, la exigencia de estabilidad.

5.2.1.4 SENSIBILIDAD A REMUESTREO

6. **Caso 6:** ídem que el caso 1, incluyendo remuestreo con los $n-1$ datos seleccionados del conjunto de datos total.

El objetivo de trabajar con distintos parámetros obtenidos a partir un muestreo con reemplazamiento de las series de datos históricos. El objetivo es introducir otra medida de distorsión completamente diferente a las introducidas en los experimentos anteriores. Para ello, se seleccionan aleatoriamente, con reemplazamiento, los meses del conjunto de las series de rendimientos mensuales, hasta completar el número total de meses existentes. Es decir, podrá seleccionarse una o más veces la serie correspondiente al mismo mes. De este modo, si durante el proceso de optimización vamos cambiando parámetros en cada generación, estaríamos considerando simultáneamente un conjunto de “escenarios posibles” para evitar hiperespecializar la solución en uno sólo, que podría ajustarse exactamente o no a la realidad. El problema de la hiperespecialización para un único escenario es que el resultado podría ser potencialmente muy sensible a pequeñas desviaciones entre el escenario previsto y el que se da en la realidad, por lo que la solución obtenida sería poco robusta.

5.2.1.5 VALIDACIÓN DE LAS TÉCNICAS PROPUESTAS

El siguiente paso consiste en verificar si mediante los cambios introducidos a la composición de la cartera, se obtienen soluciones más próximas a la realidad o no.

Se han implementado dos formas diferentes de proporcionar robustez a las carteras de inversión, tratando de introducir diferentes medidas de distorsión a las estimaciones de los parámetros calculadas.

La primera forma se encarga de trabajar con escenarios distintos a los que se tienen previstos. Consiste en modificar el funcionamiento del algoritmo, mediante la obtención de parámetros a partir de muestreo con reemplazamiento, tal y como ya se ha explicado.

La segunda forma de construir carteras robustas (combinable con la anterior), consiste en modelar de forma explícita la estabilidad, e incorporarla al sistema como un tercer objetivo a optimizar, minimizando la métrica para exigir mayor estabilidad. El modo en que se llevarán a cabo los experimentos relacionados con este objetivo, se basa en la exploración del entorno de una cartera, sin modificar los parámetros del modelo. Se trata de introducir pequeñas modificaciones a estos parámetros para simular variaciones que puedan producirse con respecto a la estimación prevista. Estamos intentando conseguir un sistema que no sea inútil ante estos errores, capaz de soportar pequeñas equivocaciones en los cálculos, sin que se produzcan grandes alteraciones en los resultados. Se perseguirá que los resultados reales se parezcan a los previstos. Para evaluar bajo qué parámetros se consigue una mayor estabilidad, se agrupa el valor de este objetivo por niveles y se hacen comparaciones entre ellos, utilizando métricas de desempeño que habrá analizar. Como estamos trabajando con errores muy pequeños, será necesario emplear contrastes estadísticos para analizar los resultados de estas métricas.

Puesto que desconocemos qué sucedió en los meses posteriores a nuestras muestras, dividimos el conjunto de datos en dos partes. El primer conjunto trabajará con $n-1$ datos, para hacer las estimaciones, mientras que el segundo evaluará qué sucedió realmente en ese

instante (para el dato n). Por lo tanto, se ejecutarán todos los casos de prueba dos veces, uno por cada conjunto de datos. El primero servirá de estimación para evaluar las dispersiones finales producidas, ayudándonos de una nueva métrica de desempeño, calculada a partir de estos errores. Sus resultados proporcionarán una medida para la estabilidad de las estimaciones (completamente distintas a las de hipervolumen y spread, nombradas). De este análisis se valorarán la utilidad del uso del modelado de los métodos de robustez propuestos, así como las configuraciones con menores alteraciones en los resultados finales.

5.2.1.6 MÉTRICAS

Para comparar los frentes que se obtengan durante la fase experimental se utilizan dos de las métricas estándar normalmente adoptadas en la literatura de optimización evolutiva multiobjetivo: las métricas *Spread* e *Hipervolumen*. Además, se considerará una tercera a la que denominaremos “Error de estimación”.

A la hora de evaluar el rendimiento de algoritmos multiobjetivo, se suele tener en cuenta dos aspectos: minimizar la distancia del frente de Pareto obtenido por el algoritmo al frente de Pareto exacto del problema y maximizar la extensión de soluciones sobre el frente, de forma que la distribución sea lo más uniforme posible. En este sentido, las métricas usadas en el campo se pueden agrupar entre aquellas que evalúan la cercanía a los frentes exactos, las que miden la diversidad de las soluciones no dominadas obtenidas, o ambas [27]. Aquí se ha empleado una de cada tipo.

Debe tenerse en mente que la evaluación comparativa de resultados entre los algoritmos genéticos multiobjetivo es un tema complejo. Para enfocar este asunto se presenta el siguiente ejemplo, tomado de *Deb* (2002). En un escenario como el presentado en la Figura 28 es posible concluir que el algoritmo *A* supera en calidad de resultados al algoritmo *B*, al mostrar en sus resultados finales mejor convergencia al frente de Pareto y aproximadamente la misma diversidad. Pero en una situación más confusa como la que se presenta en la Figura 29, las conclusiones no pueden ser formuladas ligeramente. Algunos puntos del algoritmo *A* dominan a los resultados finales del algoritmo *B*, mientras que otros son dominados. La determinación de qué algoritmo es “mejor” dependerá de la definición exacta de las métricas utilizadas. Asimismo, la naturaleza no determinística de los EAs hace necesario realizar estudios estadísticos sobre un número considerable de ejecuciones independientes para tomar decisiones sobre la superioridad de un algoritmo sobre otro en términos de calidad de soluciones obtenidas.

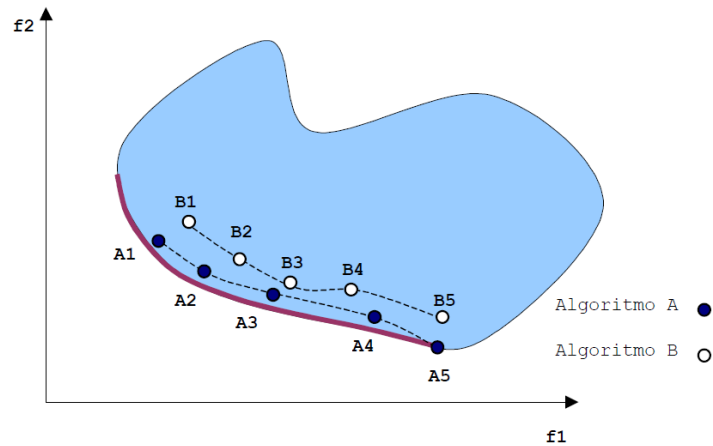


Figura 28: Ejemplo comparación de algoritmos

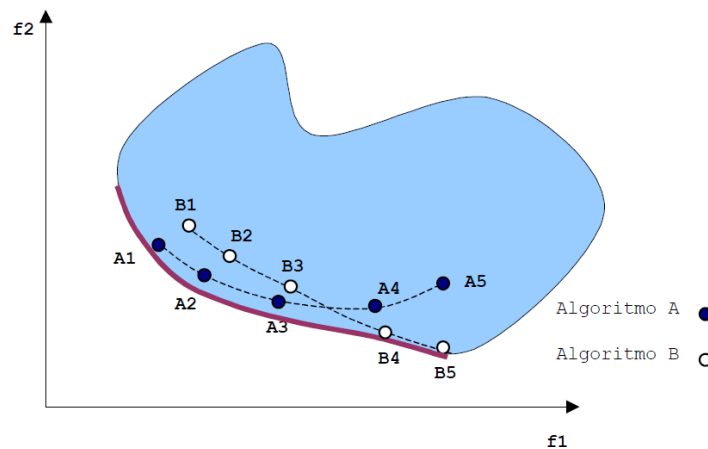


Figura 29: Ejemplo II de comparación de algoritmos

ESPACIO DISTRIBUIDO DE SCHOTT (SPREAD):

La dispersión o Δ es un indicador de diversidad que mide la distribución de las soluciones obtenidas. Esta métrica se define como:

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} |d_i - \bar{d}|}{d_f + d_l + (N - 1)\bar{d}},$$

donde d_i es la distancia Euclídea entre dos soluciones consecutivas, \bar{d} es la media de estas distancias, y d_f y d_l son las distancias euclídeas a las soluciones extremas (límite) del frente óptimo en el espacio objetivo [26]. Esta medida toma el valor cero para una distribución ideal, cuando hay una dispersión perfecta de las soluciones del frente de Pareto. Por lo tanto, cuanto más cercanos a 0 sean los valores obtenidos, mejor se considerará el frente de Pareto conseguido.

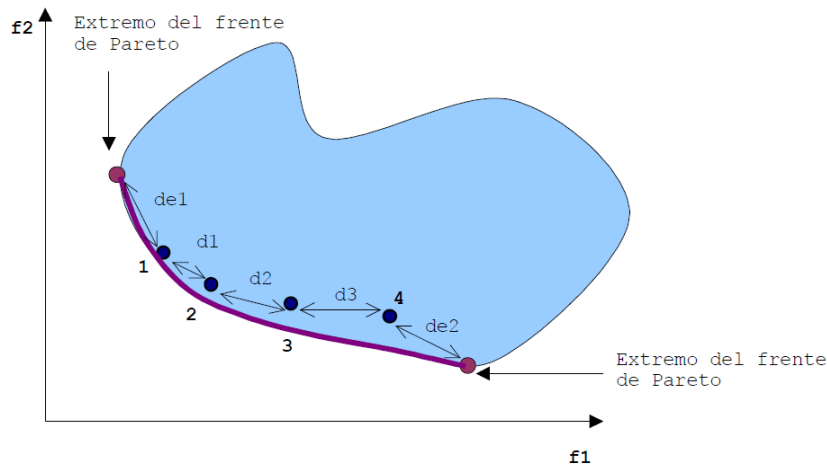


Figura 30: Ejemplo representación de la dispersión

HIPERVOLUMEN:

La métrica de *hipervolumen* (Zitzler99a) es un indicador combinado de convergencia y diversidad que calcula el volumen, en el espacio de objetivos, cubierto por los miembros de un conjunto Q de soluciones no dominadas (la región acotada por la línea discontinua en la Figura 3.11, $Q = \{A, B, C\}$) para problemas en los que todos los objetivos deben ser minimizados. Matemáticamente, para cada solución $i \in Q$, un hipercubo v_i se construye utilizando un punto de referencia W (que puede estar compuesto por la peor solución para cada objetivo, por ejemplo) y la solución i como las esquinas de la diagonal del hipercubo. El punto de referencia se puede obtener simplemente construyendo un vector de los mejores valores para las funciones. Los algoritmos que alcanzan mayores valores para HV son mejores. Así, HV se calcula como el volumen de la unión de todos los hipercubos.

$$HV = \text{volumen} \left(\bigcup_{i=1}^{|Q|} v_i \right) .$$

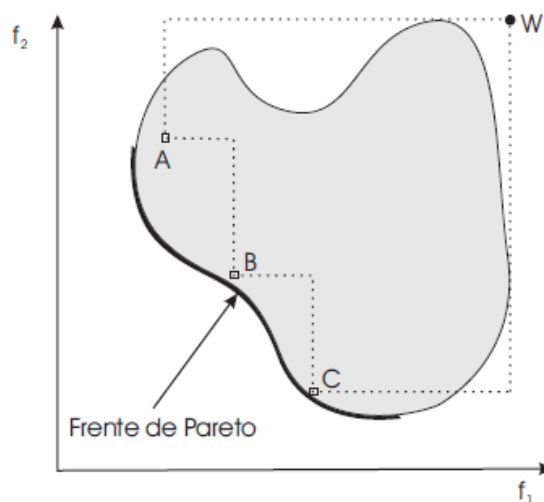


Figura 31: Hipervolumen cubierto por las soluciones no dominadas

Hay que indicar que para estas dos métricas es necesario normalizar, de alguna forma, los frentes obtenidos, para obtener resultados comparables. Esta normalización, realizada automáticamente JMetal, se hace con respecto al frente de Pareto real. En este caso, dado que no contamos con él, emplearemos en su lugar un frente de referencia compuesto a partir de las mejores soluciones obtenidas en todas las ejecuciones.

ERROR DE ESTIMACIÓN:

La evaluación de la proximidad entre el conjunto de puntos resultantes de los pares rentabilidad-riesgo para el dato n , y los frentes estimados, a partir de los $n-1$ datos anteriores, requiere la definición de una nueva métrica.

La métrica mide la diferencia promedio entre las rentabilidades y riesgos previstos y los reales. Tal y como se mencionó, los primeros se hicieron a partir de una estimación para los valores con los parámetros con la información disponible en $n-1$ datos. La métrica se calcula midiendo la distancia de Mahalanobis entre la rentabilidad y riesgo previstos para cada cartera, y la rentabilidad y riesgo de esa misma cartera, calculados con los parámetros reales (los observados en n).

La fórmula matemática de esta métrica se expresa como sigue:

$$\sum_{i=1}^n dm \frac{(\vec{x}_i, \vec{x}_i')}{n},$$

donde \vec{x}_i , expresa el par rentabilidad-riesgo previsto.

y \vec{x}_i' expresa el par rentabilidad-riesgo obtenido.

5.2.1.7 ANÁLISIS ESTADÍSTICO DE LOS RESULTADOS

Una vez definidos los indicadores de calidad y rendimiento, y realizadas las 42 ejecuciones independientes, se tiene un conjunto de datos por cada indicador. Desde el punto de vista estadístico, estos datos se pueden considerar como muestras de una función de densidad de probabilidad y , para poder sacar conclusiones correctas, se ha realizado el siguiente análisis estadístico, por medio de un script implementado en lenguaje R.

Primero se aplica un test de *Kolmogorov-Smirnov* para determinar si los valores obtenidos siguen una distribución normal (de Gauss). Si la siguen, se utiliza el test de *Levene* para comprobar la homocedasticidad de las muestras (igualdad de las varianzas). Si este test es positivo (las varianzas son iguales), se realiza un t-test; en caso contrario, se usa el test de *Welch*. Para distribuciones no gaussianas, se emplea el test no paramétrico de *Wilcoxon* para comparar las medianas de los resultados. Se han establecido dos niveles de confianza para todos los tests, al 99% y 95% (es decir, niveles de significación de 1% y 5%). La Figura 32 muestra gráficamente el procedimiento seguido:

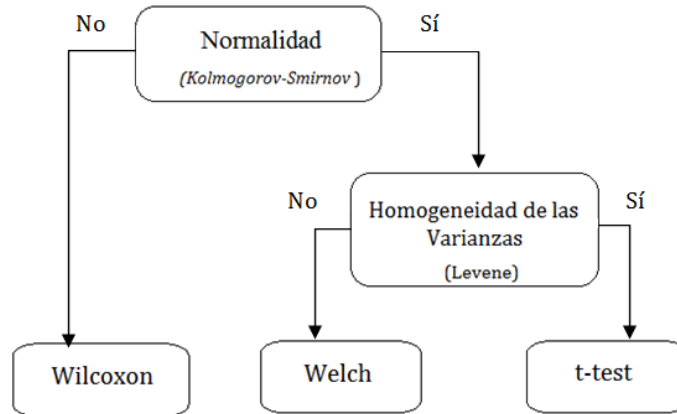


Figura 32: Análisis estadístico de los experimentos

Los resultados obtenidos en los contrastes estadísticos se recogen en tablas con el formato mostrado en la figura 33.

Formato de las tablas resultantes de los contrastes estadísticos							
	HVA	HVB	HVC		SPREADA	SPREADB	SPREADC
HVA				SPREADA			
HVB	--			SPREADB	=		
HVC	--	--		SPREADC	++	++	

Tabla 7: Formato de las tablas resultantes de los contrastes estadísticos

Los signos “++” y “--” representan una diferencia significativa al 1% entre los elementos contrastados, los signos “+” y “-” simbolizan una diferencia significativa del 5%, y el signo = quiere decir que no existen diferencias significativas entre los elementos contrastados. Los contrastes se llevan a cabo comparando el elemento de cada fila con el de cada columna, de modo que el signo “++” representa que el elemento de la fila que se está comparando es mayor con una diferencia significativa al 1% con respecto al elemento de la columna comparado. El signo “--” simboliza justo lo opuesto (el elemento de la fila es menor que el de la columna, bajo la misma diferencia significativa). Las celdas vacías se corresponden con la diagonal, donde se realizan los contrastes de las comparaciones de un elemento consigo mismo, lo que no tiene sentido. El resto de las celdas vacías representan contrastes de comparaciones ya realizadas, pero a la inversa, por lo que los resultados son los mismos que los ya obtenidos, pero con signo contrario.

5.2.1.8 AJUSTE DE PARÁMETROS Y OPERADORES

Los resultados que se presentan en el proyecto, fueron obtenidos con la configuración de parámetros que se ofrecen en la Tabla 8. Los valores de probabilidad fueron tomados de Deb et al. (2000), que son los valores utilizados por defecto en el framework y empleados para esos algoritmos. Se comprobó que estos valores tuvieran sentido en este dominio, realizando validaciones con otro software de optimización (JPortfolio, comentado en la

sección 5.1). Las soluciones se solapaban con las nuestras, por lo que se consideraron adecuados para nuestro problema.

Respecto al tamaño de las poblaciones utilizadas, éstas varían en función de la prueba realizada. Para la selección del mejor algoritmo, se propuso trabajar con un tamaño de 100 individuos, valor sugerido además en Deb et al. (2000). Teniendo en cuenta la complejidad de las funciones objetivo, se consideró que era un tamaño suficiente. Sin embargo, para las pruebas con tres objetivos, se trabajó con un mayor número de individuos (1002 en concreto), puesto que los ficheros que se obtengan tras la ejecución serán divididos en tres partes, en función del nivel de estabilidad. De éstos, a su vez, se obtendrá otro fichero con las soluciones no dominadas de cada uno de dichos niveles. Así, el fichero quedaría dividido en poblaciones de 334 individuos, en función de su estabilidad, y las soluciones no dominadas estarían compuestas por un rango que varía entre los 100 y 250 individuos, aproximadamente. Se pretende trabajar con poblaciones de tamaño similar a las utilizadas para el frente obtenido al considerar solamente los objetivos de rentabilidad y riesgo y así, poder evaluar, en similares condiciones, el rendimiento de los algoritmos.

Se definió un criterio de parada basado en especificar un esfuerzo prefijado, determinado por un número de generaciones. Al igual que ocurría con el tamaño de la población, se trabajó con dos valores distintos en función del tipo de prueba realizada. Para la composición inicial de algoritmos, el número de generaciones se fijó en 3000, mientras que para el resto de las pruebas con tres objetivos, se utilizó un número mucho mayor, 30000 generaciones.

Para estimar la sensibilidad de las carteras a pequeñas diferencias en su composición, se estableció un número de 25 carteras similares a la que se va a evaluar, parámetro fijo para todos los experimentos con tres objetivos.

Finalmente, los valores de las restricciones implementadas son los que se describieron en el apartado 5.1.1, parámetros fijos también para todos los experimentos con tres objetivos: se invierte en un total de seis de los ocho activos disponibles (restricción de cardinalidad), con unos umbrales de compra máxima y mínima del 60% y 5%, respectivamente.

Parámetro	Valor
Probabilidad de cruce	0.9
Probabilidad de Mutación	$1/N$, siendo N el número de variables de decisión en el problema
Índice de distribución para el cruce real	20
Índice de distribución para la mutación real	20
Número de generaciones utilizadas como criterio de parada	3000 y 30000 generaciones
Tamaño de la población	100 y 1002 individuos
Número de carteras similares a la evaluada	25
Restricción de cardinalidad	6 activos (de 8)

Umbral de compra máximo	60%
Umbral de compra mínimo	5%

Tabla 8: Parámetros de los algoritmos genéticos

Se realizaron experimentos no formalizados para determinar valores adecuados para los parámetros de las pruebas con tres objetivos. Se observó que para calcular el número de individuos y generaciones necesarios para que al dividir cada fichero en tres niveles de estabilidad, éstos tuvieran al menos 100 individuos, era necesario un frente con soluciones no dominadas cuatro veces mayor al obtenido con 100 individuos, sin dividir el fichero. Para el nivel de estabilidad superior se obtenía un fichero con 27 individuos, contando con un tamaño de población inicial de 100. El cálculo de las generaciones fue un poco a tanteo. Si con un número de generaciones 30 veces superior al número de individuos era suficiente para los 100 individuos, con 1002 individuos, podría valer con 30000 generaciones. Posiblemente alguna menos también hubiera bastado.

5.3 ANÁLISIS DE LOS RESULTADOS

Esta sección presenta los detalles de los experimentos comparativos realizados para evaluar la robustez en el problema de optimización de carteras de inversión, los resultados obtenidos y su análisis.

Tal y como se mencionó con anterioridad, para cada uno de los experimentos presentados se realizaron 42 ejecuciones independientes. La justificación de esto es intentar limitar la influencia del azar en los resultados. Los estadísticos descriptivos para las distintas métricas y los contrastes asociados que se facilitan en las tablas de resultados se corresponden con estas.

El primer experimento tiene por objeto confirmar si, tal y como se vio en la bibliografía consultada, SPEA2 tiene mejor rendimiento que NSGA-II en este dominio. Por esta razón, se hace una primera serie de ejecuciones que permiten comparar métricas de spread e hipervolumen. Para el resto de los experimentos, centrados en el análisis de la sensibilidad a distintos parámetros, sólo se evalúan los resultados del algoritmo SPEA2.

Los resultados se resumen en tablas donde se ofrecen los valores promedio, desviación estándar y mediana de las 42 ejecuciones para las métricas de spread e hipervolumen. Cuando se trabaja con tres objetivos, las métricas se dividen atendiendo a tres niveles de estabilidad: bajo, medio y alto. Además, incluyen los parámetros rango y exponente para facilitar el análisis de la robustez atendiendo a distintos niveles de distorsión y penalización por valores extremos y, en su caso, el efecto del remuestreo.

Tras un breve análisis de los resultados mostrados en las tablas, éstos se ilustran con gráficas de dispersión. Los datos de las gráficas hacen referencia sólo a uno de los frentes de Pareto obtenidos de las 42 ejecuciones llevadas a cabo. A continuación, se mostrarán y explicarán tablas resúmenes de los análisis estadísticos explicados, a fin de corroborar los resultados explicados e ilustrados previamente y poder establecer las primeras conclusiones en base a ellos.

Por último, se valora hasta qué punto el incorporar los dos métodos para conseguir el robustez, el tercer objetivo y el remuestreo, consiguen que las carteras de los frentes estimados usando los parámetros estimados se comporten en la práctica como se esperaba que lo hiciesen.

5.3.1. SELECCIÓN DEL MEJOR ALGORITMO

El objetivo de este primer experimento es confirmar que SPEA2 es una buena opción para abordar el problema. Para esto, compararemos su rendimiento con el que parece ser el segundo algoritmo más competitivo en el dominio, según la literatura consultada [22]. Seleccionaremos el algoritmo que mejor se ajuste a nuestro problema, con el que se realizarán el resto de los experimentos. Ambos algoritmos destacan por su eficiencia, siendo referencia en esta materia, por lo que se prevé la obtención de resultados muy similares.

Los resultados presentados en la Tabla 9 corresponden a la ejecución con una población de 100 individuos y un criterio de parada de 3000 generaciones, valores de los parámetros que permiten apreciar el potencial que presentan ambos algoritmos.

Experimento 1				
Objetivos	Algoritmo	Métricas	Medidas	Valores
2	NSGAI	Hipervolumen	Media	$7,71e^{-01}$
			Mediana	$7,75e^{-01}$
			Desviación típica	$9,3e^{-03}$
		Spread	Media	$6,42e^{-01}$
			Mediana	$6,49e^{-01}$
			Desviación típica	$3,9e^{-02}$
	SPEA2	Hipervolumen	Media	$7,80e^{-01}$
			Mediana	$7,81e^{-01}$
			Desviación típica	$6,5e^{-03}$
		Spread	Media	$6,27e^{-01}$
			Mediana	$6,26e^{-01}$
			Desviación típica	$3,6e^{-02}$

Tabla 9: Resultados primer experimento

Como puede apreciarse en la Tabla 9, ambos algoritmos han obtenido resultados de similar calidad, tanto al evaluar la aproximación al frente de Pareto, como la diversidad de las soluciones finales. Sin embargo, puede apreciarse que el algoritmo SPEA2 obtiene mejores valores tanto para el hipervolumen (el promedio de sus valores es algo más alto), como en la distribución de puntos (el valor medio de la métrica spread es más pequeño).

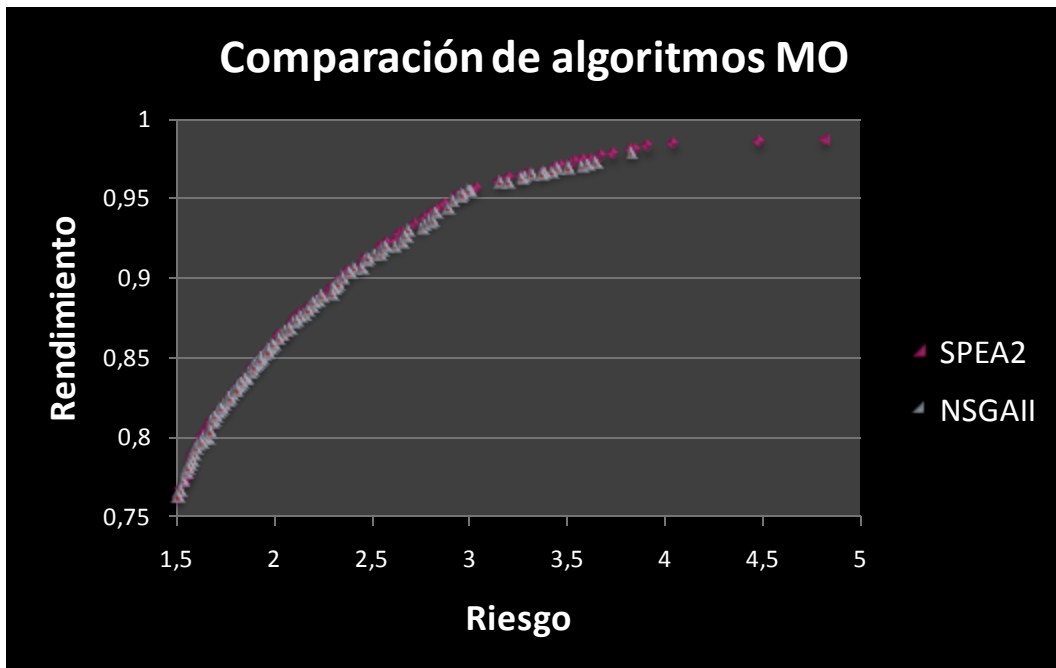


Figura 33: Comparación de los algoritmos multiobjetivo

Comparando de forma visual los frentes de Pareto obtenidos en una ejecución, se puede observar que ambos algoritmos son prácticamente idénticos, destacando en la parte superior de la curva el algoritmo SPEA2 quien, para un mismo riesgo, ofrece un mayor rendimiento, si bien la diferencia es mínima.

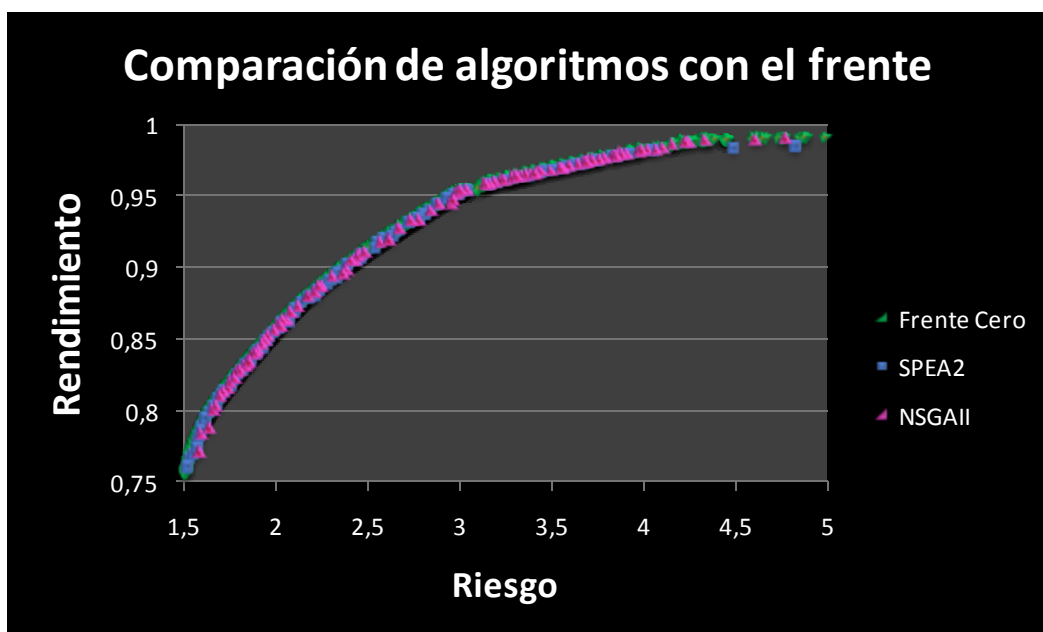


Figura 34: Comparación de los algoritmos multiobjetivo

Si se comparan ambos algoritmos con el frente de referencia calculado combinando los mejores individuos de todas las ejecuciones, figura y, puede observarse que el obtenido con SPEA2 parece aproximarse más a él, que el resultante con el algoritmo NSGA-II, lo que

lleva a concluir, observando ambos frentes, que el algoritmo SPEA2 domina en mayor grado al frente obtenido con el algoritmo NSGA-II, ofreciendo un mejor desempeño para este problema.

Atendiendo a los resultados obtenidos tras el análisis estadístico de las métricas de desempeño, puede observarse que el hipervolumen para el algoritmo SPEA2 es mayor que el del algoritmo NSGA-II, con una diferencia significativa del 1%. En cambio, en términos de spread, ambos frentes son más parecidos, distribuyendo nuevamente mejor los puntos el algoritmo SPEA2, con una diferencia significativa del 5% con respecto a la distribución realizada mediante el algoritmo NSGA-II. Por tanto, obedeciendo a los resultados obtenidos y teniendo en cuenta que la literatura del estado del arte en este área considera SPEA2 mejor algoritmo, corroborado también en las representaciones gráficas, seleccionaremos SPEA2 para la ejecución del resto de los experimentos.

	HVNSGAII	HVSPEA2		SPREADNSGAII	SPREADSPEA2
HVNSGAII			SPREADNSGAII		
HVSPEA2	++		SPREADSPEA2	-	

Tabla 10: Resultados de los contrastes de hipótesis

5.3.2 SENSIBILIDAD AL RADIO

El objetivo de esta prueba es estudiar la sensibilidad de la función objetivo de robustez propuesta al radio de distorsión aplicado sobre la composición de la cartera evaluada. Se pretende evaluar cuánto influye el nivel de esta distorsión sobre el grado de dispersión en el de objetivos y, si a un mayor rango de variación entre dichas carteras se obtienen soluciones más estables o no.

5.3.2.1. RESULTADOS GENERALES

Experimento 2						
Objetivos	Rango	Exponente	Estabilidad	Métricas	Medidas	Valores
3	5%	1	Alta	HV	Media	$7,57e^{-01}$
					Mediana	$7,57e^{-01}$
					D.típica	$3,8e^{-03}$
				Spread	Media	$7,55e^{-01}$
					Mediana	$7,54e^{-01}$
					D.típica	$3,4e^{-02}$
			Media	HV	Media	$7,79e^{-01}$
					Mediana	$7,80e^{-01}$
					D. típica	$2,4e^{-03}$
				Spread	Media	$7,88e^{-01}$
					Mediana	$7,87e^{-01}$
					D.típica	$4,6e^{-02}$
Baja	HV	Media	$7,72e^{-01}$			
		Mediana	$7,74e^{-01}$			

					D.típica	$8,5e^{-03}$
				Spread	Media	$9,97e^{-01}$
					Mediana	1,01
					D.típica	$7,0e^{-02}$
3	10%	1	Alta	HV	Media	$7,46e^{-01}$
					Mediana	$7,47e^{-01}$
					D.típica	$4,4e^{-03}$
				Spread	Media	$7,37e^{-01}$
					Mediana	$7,37e^{-01}$
					D.típica	$3,0e^{-02}$
			Media	HV	Media	$7,76e^{-01}$
					Mediana	$7,76e^{-01}$
					D. típica	$2,8e^{-03}$
				Spread	Media	$7,97e^{-01}$
					Mediana	$7,93e^{-01}$
					D.típica	$3,4e^{-02}$
Baja	HV	Media	$7,85e^{-01}$			
		Mediana	$7,85e^{-01}$			
		D.típica	$2,8e^{-03}$			
	Spread	Media	$8,70e^{-01}$			
		Mediana	$8,61e^{-01}$			
		D.típica	$4,6e^{-02}$			
3	15%	1	Alta	HV	Media	$7,25e^{-01}$
					Mediana	$7,26e^{-01}$
					D.típica	$5,9e^{-03}$
				Spread	Media	$7,44e^{-01}$
					Mediana	$7,42e^{-01}$
					D.típica	$3,8e^{-02}$
			Media	HV	Media	$7,71e^{-01}$
					Mediana	$7,71e^{-01}$
					D. típica	$3,6e^{-03}$
				Spread	Media	$8,31e^{-01}$
					Mediana	$8,31e^{-01}$
					D.típica	$2,5e^{-02}$
Baja	HV	Media	$7,85e^{-01}$			
		Mediana	$7,86e^{-01}$			
		D.típica	$2,1e^{-03}$			
	Spread	Media	$8,71e^{-01}$			
		Mediana	$8,71e^{-01}$			
		D.típica	$3,9e^{-02}$			

Tabla 11: Resultados segundo experimento

Observando los resultados, puede comprobarse cómo los valores de hipervolumen tienden a disminuir suavemente cuanto mayor es el radio, para las estabilidades alta y media. No obstante, para la estabilidad baja no sólo no se cumple esta regla sino que los valores más altos se consiguen para un rango de variación del 10% y 15%. Los valores de

hipervolumen decrecen inversamente proporcional al nivel de estabilidad exigido, tal y como se esperaba.

Los valores de spread expresan una baja calidad en la distribución de los puntos de los frentes de Pareto obtenidos. Su valor parece empeorar cuanto menor es la estabilidad y mejorar directamente proporcional al crecimiento de los porcentajes de los rangos de variación. Excepto para el nivel de estabilidad media que ocurre exactamente lo contrario, obteniendo su peor valor con un rango de variación del 10%. Los mejores valores se obtienen para el máximo nivel de estabilidad, con un 10% del rango de variación.

5.3.2.2. RESULTADOS AGRUPANDO POR RADIO

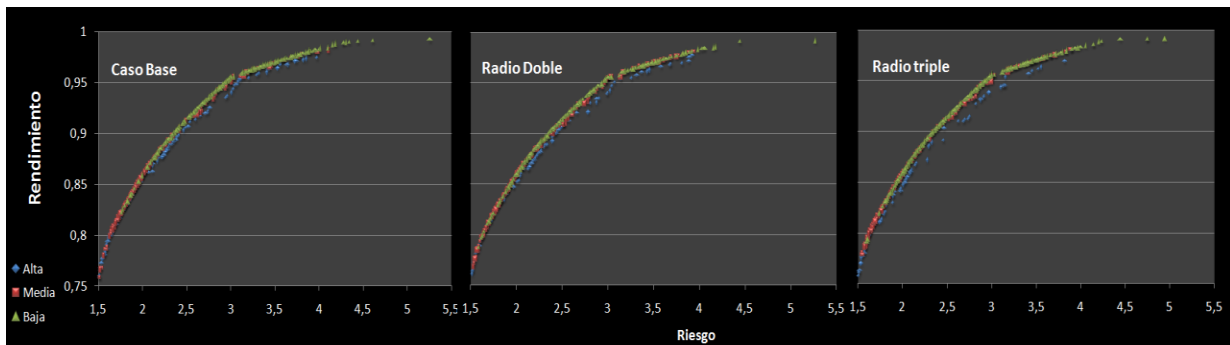


Figura 35: Gráfica comparativa de los resultados en función del radio

Agrupando los niveles en una sola gráfica para cada porcentaje del rango de variación de la estabilidad establecido, puede apreciarse como el frente de estabilidad baja domina siempre a los otros dos frentes, quedando siempre dominado el de estabilidad alta por los otros. Se hace evidente el coste existente derivado de la robustez, en términos de rentabilidad riesgo, aunque como puede observarse, la diferencia es mínima. Es destacable la similitud existente entre las tres gráficas, pudiendo verificarse una mejoría en la distribución de los puntos del frente para el rango de variación triple, al 15%.

A continuación se muestran las tablas correspondientes a los contrastes asociados a las gráficas anteriores:

Caso Base							
	HVAlta	HVBaja	HVMedia		SPREADAlta	SPREADBaja	SPREADMedia
HVAlta				SPREADAlta			
HVBaja	++			SPREADBaja	++		
HVMedia	++	++		SPREADMedia	++	--	

Tabla 12: Análisis estadístico caso base

Radio Doble							
	HVAlta	HVBaja	HVMedia		SPREADAlta	SPREADBaja	SPREADMedia
HVAlta				SPREADAlta			
HVBaja	++			SPREADBaja	++		
HVMedia	++	--		SPREADMedia	++	--	

Tabla 13: Análisis estadístico radio doble

Radio triple							
	HVAlta	HVBaja	HVMedia		SPREADAlta	SPREADBaja	SPREADMedia
HVAlta				SPREADAlta			
HVBaja	++			SPREADBaja	++		
HVMedia	++	--		SPREADMedia	++	--	

Tabla 14: Análisis estadístico radio triple

Las tablas reflejan resultados prácticamente idénticos para las tres configuraciones. Como se ha mencionado, el hipervolumen mejora cuanto menor es la estabilidad (a excepción del caso base, en el que la estabilidad media obtiene un mejor valor). Es decir, los mayores valores de la medida de dispersión del hipervolumen se obtienen para niveles de estabilidad bajos, con una diferencia significativa del 1% con respecto al hipervolumen del nivel de estabilidad inmediatamente superior. En cuanto a la distribución de los puntos, ocurre exactamente lo mismo para los tres casos: el spread es mejor cuanto mayor es la estabilidad. Por lo tanto, tal y como se preveía, para una estabilidad baja se obtienen los mejores frentes de Pareto en términos de rentabilidad-riesgo, viéndose afectados estos valores negativamente de manera proporcional a la estabilidad exigida en el sistema.

5.3.2.3. RESULTADOS AGRUPANDO POR ESTABILIDAD

La figura siguiente presenta gráficamente los puntos no dominados para cada uno de los porcentajes del rango de variación considerados, agrupados por niveles de estabilidad.

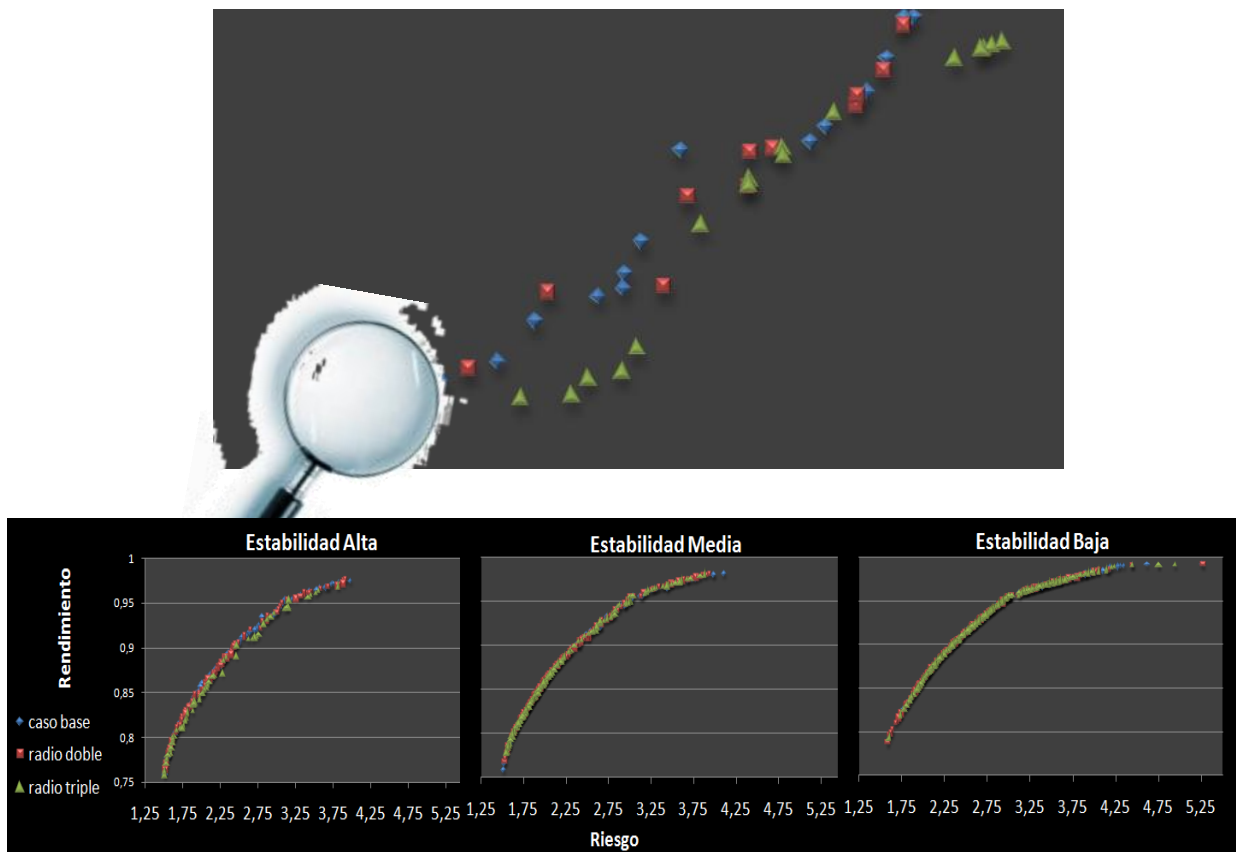


Figura 36: Análisis estadístico agrupado por estabilidades

Puede apreciarse la similitud de los frentes obtenidos, llegándose a solapar los tres para la estabilidad baja y media. En cambio, en los frentes de Pareto obtenidos con estabilidad alta se pueden distinguir los tres colores correspondientes a los distintos porcentajes. Observando la zona ampliada del gráfico, puede verse como los rangos al 5 y 10% parecen dominar al final de la curva al rango del 15%. Vuelve a confirmarse que cuanto mayor es la estabilidad exigida por el sistema, mayor es el coste en términos de rentabilidad-riesgo obtenido, pues como puede verse, cuanto menor es el rango de variación, mayor es la dominancia de Pareto conseguida.

Ayudándonos con contrastes estadísticos, se intentarán verificar las conclusiones extraídas de las representaciones gráficas de los frentes de Pareto.

Estabilidad alta							
	HVC.Base	HVR.Doble	HVR.Triple		SPREADC.Base	SPREADR.Doble	SPREADR.Triple
HVC.Base				SPREADC.Base			
HVR.Doble	--			SPREADR.Doble	--		
HVR.Triple	--	--		SPREADR.Triple	=	=	

Tabla 15: Análisis estadístico estabilidad alta

Estabilidad media							
	HVC.Base	HVR.Doble	HVR.Triple		SPREADC.Base	SPREADR.Doble	SPREADR.Triple
HVC.Base				SPREADC.Base			
HVR.Doble	--			SPREADR.Doble	=		
HVR.Triple	--	--		SPREADR.Triple	++	++	

Tabla 16: Análisis estadístico estabilidad media

Estabilidad baja							
	HVC.Base	HVR.Doble	HVR.Triple		SPREADC.Base	SPREADR.Doble	SPREADR.Triple
HVC.Base				SPREADC.Base			
HVR.Doble	++			SPREADR.Doble	--		
HVR.Triple	++	=		SPREADR.Triple	--	=	

Tabla 17: Análisis estadístico estabilidad baja

Las dos primeras tablas son idénticas en términos de hipervolumen, y contrarias con respecto a los resultados obtenidos para la estabilidad baja. Para esta última, los frentes de Pareto más próximos al óptimo se obtienen con el rango de variación al 10% y al 15%, pues sus valores de hipervolumen son mayores, con una diferencia significativa del 1%, con respecto a los valores obtenidos en esta misma medida de dispersión para el caso base.

Observando de nuevo las dos primeras tablas, se demuestra el coste de la estabilidad en términos de rendimiento-riesgo, pues los valores de hipervolumen obtenidos son peores, con una diferencia significativa del 1%, cuanto mayor es el rango de variación.

En cuanto a la distribución de los puntos, para un nivel de estabilidad medio y un rango de variación del 15%, el spread empeora, con un nivel significativo del 1%, con respecto a los valores obtenidos para unos rangos del 5% y 10%. Para un nivel de estabilidad baja, los valores de spread óptimos se obtienen para el caso base, con una diferencia significativa del 1% con respecto a los rangos de variación del 10% y 15% (ambos considerados iguales, con una distribución de puntos sin diferencia significativa aparente). Finalmente, para el nivel de estabilidad más alto, el tamaño del radio parece influir poco en la distribución de puntos de los frentes de Pareto. Por lo tanto, puede confirmarse que cuanto mayor es el porcentaje del rango de variación (o radio), más estable es la solución obtenida (excepto para la solución más inestable, que parece ocurrir exactamente lo opuesto). La distribución de los puntos se ve poco afectada por el porcentaje del radio analizado.

5.3.3. SENSIBILIDAD AL EXPONENTE

El objetivo de esta prueba, similar a la anterior, es analizar el exponente óptimo como medida de penalización para aquellas carteras creadas a similitud de la evaluada que más difiera respecto a ésta. Teóricamente, cuanto mayor sea el exponente al que se eleve la distancia de Mahalanobis obtenida, mayor será la penalización, y por tanto, la exigencia de que no se dan desviaciones extremas.

5.3.3.1. RESULTADOS GENERALES

Experimento 3							
Objetivos	Rango	Exponente	Estabilidad	Métricas	Medidas	Valores	
3	5%	1	Alta	HV	Media	$7,57e^{-01}$	
					Mediana	$7,57e^{-01}$	
					D.típica	$3,8e^{-03}$	
				Spread	Media	$7,55e^{-01}$	
					Mediana	$7,54e^{-01}$	
					D.típica	$3,4e^{-02}$	
			Media	HV	Media	$7,79e^{-01}$	
					Mediana	$7,80e^{-01}$	
					D. típica	$2,4e^{-03}$	
				Spread	Media	$7,88e^{-01}$	
					Mediana	$7,87e^{-01}$	
					D.típica	$4,6e^{-02}$	
				Baja	HV	Media	$7,72e^{-01}$
						Mediana	$7,74e^{-01}$
						D.típica	$8,5e^{-03}$
Spread	Media	$9,97e^{-01}$					
	Mediana	1,01					
	D.típica	$7,0e^{-02}$					
3	5%	2	Alta	HV	Media	$7,52e^{-01}$	
					Mediana	$7,53e^{-01}$	

					D.típica	$2,51e^{-05}$			
				Spread	Media	$7,86e^{-01}$			
					Mediana	$7,84e^{-01}$			
					D.típica	$1,25e^{-03}$			
			Media	HV	Media	$7,76e^{-01}$			
						Mediana	$7,76e^{-01}$		
						D. típica	$1,32e^{-05}$		
			Baja	Spread	Media	$8,70e^{-01}$			
						Mediana	$8,84e^{-01}$		
						D.típica	$2,20e^{-03}$		
			Alta	HV	Media	$7,77e^{-01}$			
						Mediana	$7,78e^{-01}$		
						D.típica	$3,76e^{-05}$		
			Media	Spread	Media	$9,70e^{01}$			
						Mediana	$9,77e^{01}$		
						D.típica	$4,91e^{-03}$		
3	5%	3	Alta	HV	Media	$7,51e^{-01}$			
						Mediana	$7,52e^{-01}$		
						D.típica	$2,67e^{-05}$		
					Spread	Media	$8,01e^{-01}$		
						Mediana	$8,16e^{-01}$		
						D.típica	$1,03e^{-02}$		
					Media	HV	Media	$7,76e^{-01}$	
								Mediana	$7,76e^{-01}$
								D. típica	$1,39e^{-05}$
					Baja	Spread	Media	$8,61e^{-01}$	
								Mediana	$8,60e^{-01}$
								D.típica	$1,42e^{-03}$
		Alta	HV	Media	$7,76e^{-01}$				
					Mediana	$7,77e^{-01}$			
					D.típica	$6,77e^{-05}$			
		Media	Spread	Media	$9,55e^{01}$				
					Mediana	$9,50e^{01}$			
					D.típica	$5,40e^{-03}$			

Tabla 18: Resultados tercer experimento

Los resultados reflejan muy poca variación en términos de hipervolumen con respecto al incremento del exponente, para cada uno de los niveles de estabilidad. Para el caso base se obtienen los mejores valores de hipervolumen y spread, lo cual es lógico. Cuanto menor sea la perturbación introducida en el sistema, menor es la estabilidad exigida y, por tanto, mayor su proximidad al frente de referencia. Lo mismo sucede si atendemos a los valores de hipervolumen y spread para cada nivel de estabilidad. A medida que el sistema es más inestable, mayor es la similitud del frente obtenido con respecto al frente de referencia. Los niveles de estabilidad medio y bajo permanecen invariables ante el aumento de la penalización introducida. Los niveles con mayor estabilidad son los únicos que presentan una suave sensibilidad al exponente.

5.3.3.2. RESULTADOS AGRUPANDO POR EXPONENTE

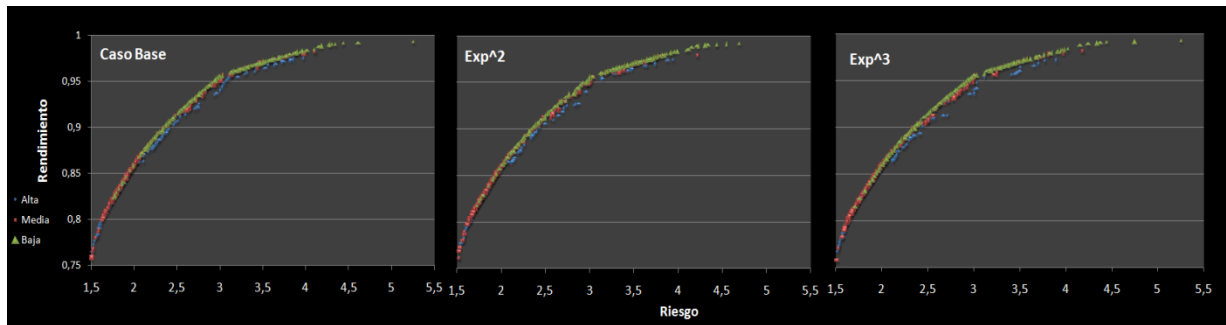


Figura 37: Análisis gráfico

Cada gráfica agrupa los niveles de estabilidad en función del exponente evaluado, permitiendo observar, nuevamente, el coste existente para la estabilidad alta, con respecto a la dominancia de los frentes de los otros dos niveles. A pesar de la proximidad entre los frentes de menor estabilidad, puede distinguirse la dominancia del frente más inestable con respecto a los otros dos. El frente con estabilidad más alta se encuentra dominado por aquellos con menor estabilidad que él.

Los contrastes confirman la relevancia estadística de las diferencias mostradas la tabla 18, y en los gráficos anteriores. Los resultados reflejan que, en términos de hipervolumen, el frente de Pareto correspondiente a la estabilidad media domina a los otros dos frentes (con una diferencia significativa del 1%), siendo el peor de ellos el de mayor estabilidad. En lo relativo a la distribución de puntos, el spread se distribuye mejor cuanto mayor es el nivel de estabilidad exigido. Se obtienen mayores valores de spread a medida que disminuye la estabilidad, con una diferencia significativa del 1% entre cada nivel con el inmediatamente inferior). Por lo tanto, en contra de lo que se esperaba, el frente dominante no es el más inestable, sino el de estabilidad media. No parece influir el coste de la robustez en términos rentabilidad-riesgo para este nivel de estabilidad exigido, aunque sí para el siguiente nivel con mayor estabilidad, donde comienza a verse reflejado este coste.

Caso base							
	HVAlta	HVBaja	HVMedia		SPREADAlta	SPREADBaja	SPREADMedia
HVAlta				SPREADAlta			
HVBaja	++			SPREADBaja	++		
HVMedia	++	++		SPREADMedia	++	--	

Tabla 19: Análisis estadístico caso base

Exponente al cuadrado							
	HVAlta	HVBaja	HVMedia		SPREADAlta	SPREADBaja	SPREADMedia
HVAlta				SPREADAlta			
HVBaja	++			SPREADBaja	++		
HVMedia	++	++		SPREADMedia	++	--	

Tabla 20: Análisis estadístico exponente al cuadrado

Exponente al Cubo							
	HVAlta	HVBaja	HVMedia		SPREADAlta	SPREADBaja	SPREADMedia
HVAlta				SPREADAlta			
HVBaja	++			SPREADBaja	++		
HVMedia	++	++		SPREADMedia	++	--	

Tabla 21: Análisis estadístico exponente al cubo

5.3.3.3. RESULTADOS AGRUPANDO POR ESTABILIDAD

A continuación se representan gráficamente los puntos no dominados para cada uno de los exponentes considerados, agrupados por niveles de estabilidad.

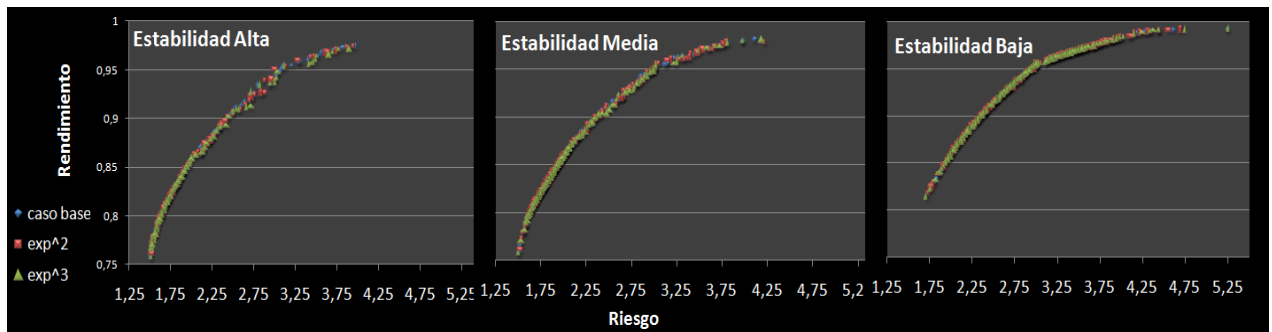


Figura 38: Análisis gráfico agrupados por estabilidad

Al observar las gráficas, podemos ver que esta medida de penalización para discriminar las distancias más grandes entre las carteras originales y las distorsionadas, tiene un efecto muy parecido al incremento del rango de variación entre ambas. Si se centra la atención en la gráfica con mayor estabilidad, puede apreciarse cómo se acentúan las distancias entre los tres frentes, pudiéndose distinguir los colores de cada uno de ellos. Sin embargo, es difícil determinar cuál de los tres exponentes es el dominante. En los frentes con menor estabilidad, parece clara la dominancia del frente obtenido al elevar el exponente al cuadrado, sobre el obtenido cuando se eleva al cubo.

Las siguientes tablas muestran los contrastes asociados a estas gráficas:

Estabilidad Alta							
	HVC.Base	HVExp^2	HVExp^3		SPREADC.Base	SPREADExp^2	SPREADExp^3
HVC.Base				SPREADC.Base			
HVExp^2	++			SPREADExp^2	--		
HVExp^3	=	--		SPREADExp^3	--	=	

Tabla 22: Análisis estadístico estabilidad alta

Estabilidad Media							
	HVC.Base	HVExp^2	HVExp^3		SPREADC.Base	SPREADExp^2	SPREADExp^3
HVC.Base				SPREADC.Base			
HVExp^2	++			SPREADExp^2	-		
HVExp^3	+	=		SPREADExp^3	--	=	

Tabla 23: análisis estadístico estabilidad media

Estabilidad Baja							
	HVC.Base	HVExp^2	HVExp^3		SPREADC.Base	SPREADExp^2	SPREADExp^3
HVC.Base				SPREADC.Base			
HVExp^2	-			SPREADExp^2	=		
HVExp^3	-	=		SPREADExp^3	=	=	

Tabla 24: Análisis estadístico estabilidad baja

Para las estabildades alta y media, las métricas de desempeño reflejan resultados similares, mientras que la solución más inestable demuestra poca sensibilidad a la variación del exponente.

Cuando se eleva el exponente al cuadrado en las dos estabildades más altas, se obtienen medidas de dispersión de hipervolumen mayores, por lo que sus frentes de Pareto se sitúan más próximos al frente de referencia utilizado. El exponente al cubo no presenta diferencias significativas, con respecto al caso base, para niveles altos de estabilidad. Sin embargo, sí que se confirma la superioridad del exponente al cuadrado frente a los anteriores. El frente resultante de elevar al cubo el exponente, con un nivel de estabilidad media, es mejor (con una diferencia significativa al 5%) que si no se eleva a nada, e idéntico que si se eleva al cuadrado.

En términos de spread, tanto para la estabilidad alta como para la media, se obtienen valores menores para el caso base, con diferencias significativas entre el 1% y el 5%, por lo que su distribución de puntos es mejor. En cambio, estas distribuciones son idénticas para el exponente al cubo y al cuadrado, al indicar el contraste que no existen diferencias significativas entre ambas.

Como ya se ha comentado, la estabilidad baja es muy poco sensible al exponente, siendo prácticamente todos los frentes de la misma dominancia (se obtienen soluciones dominantes con el frente correspondiente al caso base, con una diferencia significativa al 5% con respecto a los otros frentes), y con las mismas distribuciones de puntos.

Por lo tanto, puede confirmarse que los mejores frentes de Pareto se obtienen al elevar el exponente al cuadrado y al cubo. Sin embargo, no queda clara la contribución ofrecida por el exponente en términos de estabilidad. Es obvio que afecta a las soluciones resultantes con estabildades altas, pero no en la medida prevista, puesto que parece obtenerse mayor estabilidad cuando las soluciones no se elevan a ningún número.

5.3.4. SENSIBILIDAD AL REMUESTREO

El objetivo de este apartado es introducir una forma de distorsión completamente diferente a las introducidas en los experimentos anteriores. Se pretende optimizar simultáneamente un conjunto de “escenarios posibles”, evitando hiperespecializar la solución en uno sólo, que podría ajustarse exactamente o no a la realidad.

5.3.4.1. RESULTADOS GENERALES

Tal y como se explicó con anterioridad, la técnica del remuestreo consiste en variar en cada generación los parámetros del modelo. Esto tiene como efecto que no se cuente con un frente ideal estable. Dado que JMetal escala los ejes en función del frente ideal para el cálculo de las métricas, esta cuestión tiene importancia a la hora de evaluar los resultados.

Hasta ahora, se ha usado un frente de referencia común en todos los experimentos. Desafortunadamente, hay soluciones de los frentes obtenidos por remuestreo cuya rentabilidad o riesgo son superiores o inferiores a las máximas o mínimas del frente mencionado. Es por esta razón por la que se ha decidido calcular un nuevo frente de referencia específico para este caso. Este frente se compondrá solamente de dos puntos, la rentabilidad mínima y el riesgo mínimo obtenido en todos los frentes obtenidos por remuestreo y la rentabilidad máxima y el riesgo máximo.

Se espera que el frente con los dos puntos extremos facilite un escalado que permita comparar de forma consistente todos los resultados que se han obtenido usando esta técnica. Dicho esto, es importante recalcar que, al usar frentes de referencia distintos, los valores de estas métricas no son directamente comparables con las de los experimentos anteriores:

A continuación se resumen los resultados obtenidos al introducir remuestreo:

Experimento 4							
Objetivos	Remuestreo	Estabilidad	Métricas	Medidas	Valores		
3	No	Alta	HV	Media	$7,57e^{-01}$		
				Mediana	$7,57e^{-01}$		
				D.típica	$3,8e^{-03}$		
			Spread	Media	$7,55e^{-01}$		
				Mediana	$7,54e^{-01}$		
				D.típica	$3,4e^{-02}$		
		Media	HV	Media	$7,79e^{-01}$		
				Mediana	$7,80e^{-01}$		
				D. típica	$2,4e^{-03}$		
			Spread	Media	$7,88e^{-01}$		
				Mediana	$7,87e^{-01}$		
				D.típica	$4,6e^{-02}$		
			Baja	HV	Media	$7,72e^{-01}$	
					Mediana	$7,74e^{-01}$	
					D.típica	$8,5e^{-03}$	
Spread	Media	$9,97e^{-01}$					
	Mediana	1,01					
	D.típica	$7,0e^{-02}$					
3	Sí	Alta	HV	Media	$3,7e^{-01}$		
				Mediana	$3,62e^{-01}$		
			Spread	D.típica	$2,54e^{-04}$		
				Media	$8,84e^{-01}$		
						Mediana	$3,87e^{-01}$

				D.típica	$1.04e^{-02}$
		Media	HV	Media	$3.74e^{-01}$
				Mediana	$2.40e^{-01}$
			Spread	D. típica	$2,68e^{+09}$
				Media	$9.35e^{-01}$
		Baja	HV	Mediana	$8.72e^{-01}$
				D.típica	3.4^{-03}
			Spread	Media	$3.87e^{-01}$
				Mediana	9.21^{-01}
			HV	D.típica	$3.6e^{-03}$
				Media	$9.31e^{-01}$
		Spread	Mediana	$9.16e^{-01}$	
			D.típica	$3.4e^{-03}$	

Tabla 25: Resultados cuarto experimento

5.3.4.2. ANÁLISIS AGRUPANDO POR ESTABILIDAD

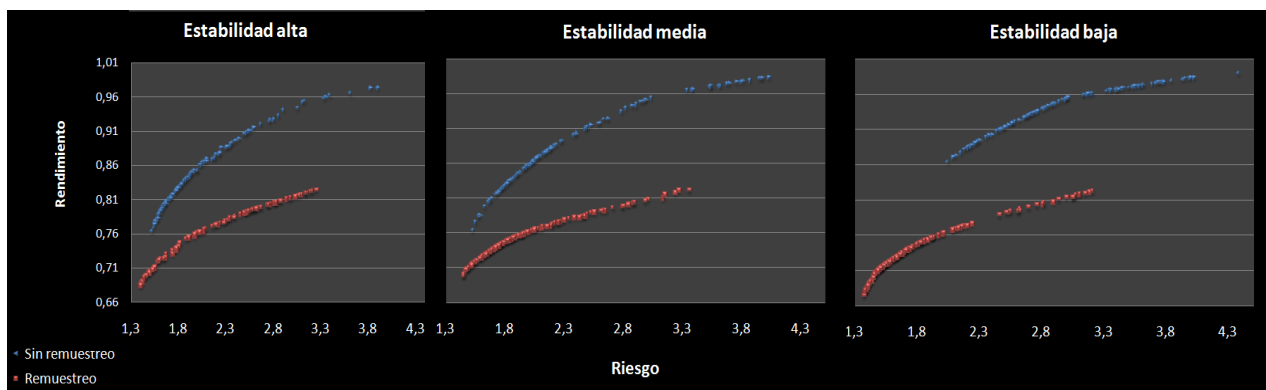


Figura 39: Análisis gráfico cuarto experimento

Las gráficas muestran claramente, independientemente del nivel de estabilidad analizado, el poco parecido existente entre ambos frentes de Pareto. Son soluciones completamente diferentes, dominadas casi en su totalidad por los frentes resultantes de la ejecución sin remuestreo.

Atendiendo, tanto a los resultados estadísticos de la tabla 25, como a estas representaciones gráficas, puede apreciarse cómo el hipervolumen y la distribución de los puntos mejoran cuanto menor es la exigencia de la estabilidad evaluada. El frente de Pareto de menor estabilidad es el mejor distribuido y más parecido, en cuanto a forma se refiere, al frente obtenido sin remuestreo.

Remuestreo (AI) vs Caso base													
	HV AI Alta	HV AI Baja	HV AI Media	HV Alta	HV Baja	HV Media		SPRE AD AI Alta	SPRE AD AI Baja	SPRE AD AI Media	SPREAD Alta	SPREAD Baja	SPREAD Media
HV AI Alta							SPREA D AI Alta						
HV AI Baja	++						SPREA D AI Baja	++					
HV AI Media	++	--					SPREA D AI Media	++	=				
HV Alta	++	--	--				SPREA D AI Alta	--	--	--			
HV Baja	=	--	--	--			SPREA D Baja	++	++	++	++		
HV Media	=	--	--	--	=		SPREA D Media	--	--	--	++	--	

Tabla 26: Estadísticos Remuestreo frente a caso base

Los contrastes confirman, claramente, los resultados mostrados en las gráficas y en la tabla 25. El valor del hipervolumen en el caso base, para todos sus niveles de estabilidad, muestra diferencias significativas con respecto al hipervolumen de los datos con remuestreo.

La distribución de los puntos del frente de Pareto para las estabilidades más altas del caso base, muestran también diferencias significativas al 1% con respecto a los valores distorsionados por remuestreo. Solo el nivel más inestable del caso base resulta distribuir los puntos peor que cualquier valor remuestreado.

5.3.5. VALIDACIÓN DE LAS TÉCNICAS PROPUESTAS

En este apartado pretendemos comprobar hasta qué punto las dos técnicas propuestas para obtener frentes robustos, el tercer objetivo y el remuestreo, resultan de utilidad. Para ello, mediremos las rentabilidades y riesgos de la cartera de las desviaciones promedio entre los frentes estimados obtenidos con los parámetros de distorsión, y el comportamiento de estas mismas carteras con los valores observados.

Se pretende analizar la influencia del radio y del exponente sobre la estabilidad, tomando nuevamente como referencia el caso base descrito al principio del capítulo.

A continuación, se comparan las distancias de Mahalanobis promedio entre los frentes estimados y las soluciones obtenidas en la realidad, recurriendo a la métrica de *Error de estimación*, definida formalmente en el apartado [5.1.1.6](#).

5.3.5.1. RESULTADOS CON DIFERENTES EXPONENTES Y NÚMERO DE OBJETIVOS

Objetivos	Medidas			Valores	
2	Media			15,11	
	Mediana			15,17	
	Desviación típica			1,67	
Objetivos	Rango	Exponente	Estabilidad	Medidas	Valores
3	5%	1	Alta	Media	12,86
				Mediana	12,76
				Desviación típica	0,60
			Media	Media	17,43
				Mediana	17,44
				Desviación típica	0,92
			Baja	Media	19,80
				Mediana	20,28
				Desviación típica	2,36
3	5%	2	Alta	Media	12,94
				Mediana	12,90
				Desviación típica	1,11
			Media	Media	12,90
				Mediana	12,82
				Desviación típica	0,67
			Baja	Media	13,05
				Mediana	13,21
				Desviación típica	1,15
3	5%	3	Alta	Media	12,80
				Mediana	12,56
				Desviación típica	0,90
			Media	Media	12,70
				Mediana	12,58
				Desviación típica	0,70
			Baja	Media	13,12
				Mediana	13,26
				Desviación típica	0,96

Tabla 27: Resultados experimento diferente número de objetivos

Los resultados obtenidos son menores cuanto mayor es la estabilidad. Empleando sólo dos objetivos, pueden apreciarse mayores diferencias entre el frente estimado y las soluciones reales, que las distancias obtenidas cuando se aplica un nivel de estabilidad alto, cualquiera que sea el exponente al que se eleve la distancia. Parecen conseguirse menores alteraciones en los resultados estimados que hacen uso de la estabilidad. Los resultados demuestran una mínima sensibilidad al exponente, alcanzándose los valores óptimos al elevarlo al cubo. No obstante, es necesario realizar contrastes de estos resultados con la nueva métrica, para confirmar los resultados mostrados en la tabla 27.

Análisis de la robustez con el exponente										
	BaseAlta	BaseBaja	BaseMed	2Obj	Exp ² Alta	Exp ² Baja	Exp ² Med	Exp ³ Alta	Exp ³ Baja	Exp ³ Med
BaseAlta										
BaseBaja	++									
BaseMed	++	--								
2 Obj	++	--	--							
Exp ² Alta	=	--	--	--						
Exp ² Baja	=	--	--	--	=					
Exp ² Med	=	--	--	--	=	=				
Exp ³ Alta	=	--	--	--	=	=				
Exp ³ Baja	=	--	--	--	=	=	=	=		
Exp ³ Med	=	--	--	--	=	-	=	=	-	

Tabla 28: análisis de la robustez con el exponente

Los contrastes realizados con esta nueva métrica corroboran los resultados estadísticos obtenidos, confirmando la influencia de la estabilidad en la mejora de las estimaciones realizadas para el dato n . Como puede verse, los errores de estimación para los niveles de estabilidad más bajos, o cuando se trabaja con dos objetivos, es mayor que para las configuraciones con niveles de estabilidad alto. El contraste indica una diferencia significativa al 1% entre los valores con estabilidad alta y los obtenidos con dos objetivos. Puede afirmarse que los frentes resultantes de la ejecución con dos objetivos, tienen una mayor distancia promedio desde los puntos del frente previsto al objetivo. Por lo tanto, los frentes con un nivel de estabilidad alto se consideran soluciones más robustas a los cambios en los valores de los parámetros, mientras que la solución obtenida con tan solo dos objetivos lo es en menor medida.

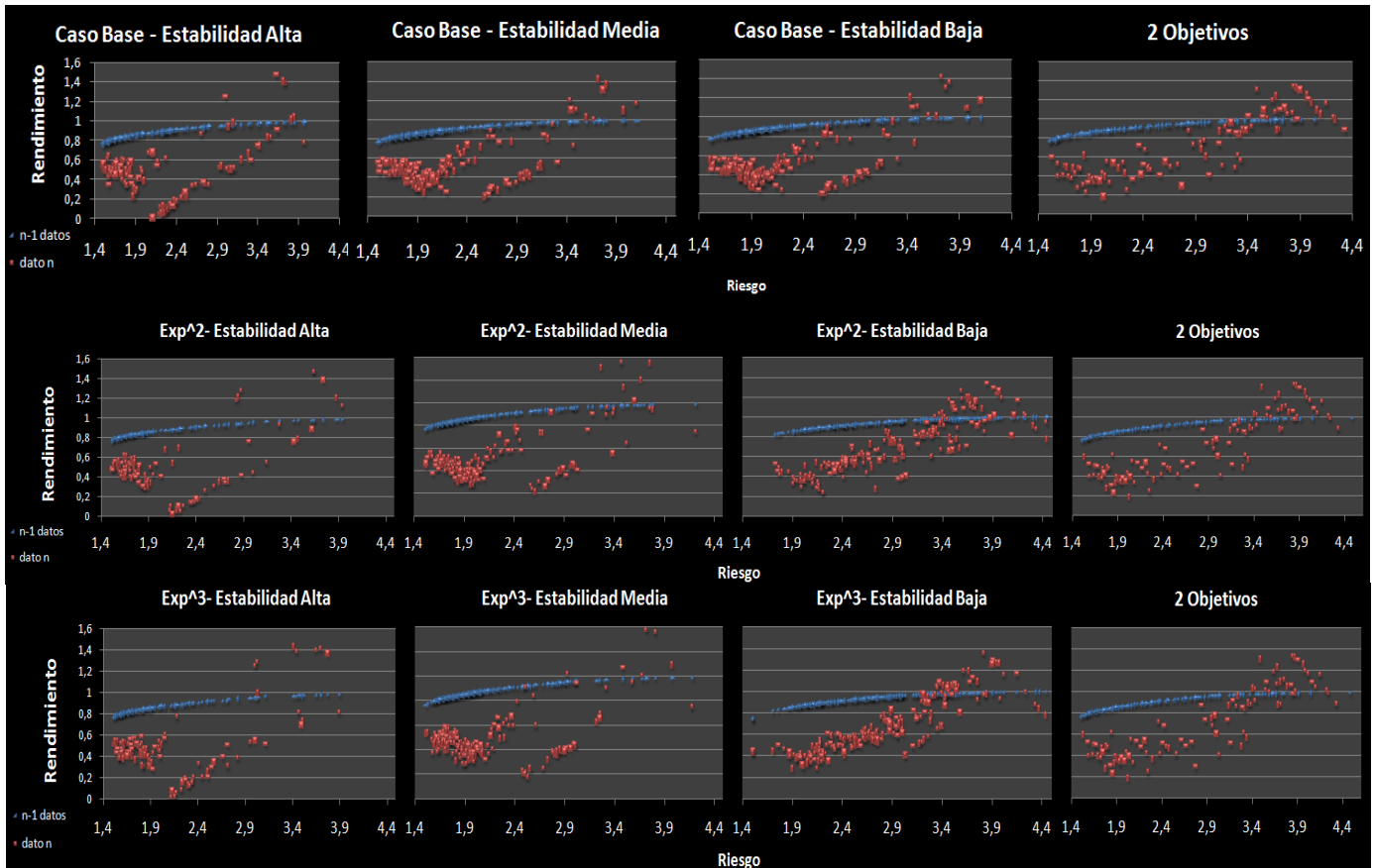


Figura 40: análisis gráfico diferentes estabildades y número de objetivos

Puede observarse una distribución de puntos muy parecida en función de los niveles de estabilidad, sin tener en cuenta el exponente. Cuanto menor es la estabilidad, mayor es la concentración de dichos puntos en torno al frente de referencia estimado (de color azul). A medida que aumenta la estabilidad, éstos parecen alejarse más del frente azul, y dispersarse por todo el espacio. Los gráficos parecen indicar lo contrario a lo que sugieren los estadísticos, pero la explicación resulta sencilla. Las soluciones reales más próximas a los frentes estimados, en realidad se corresponden con los puntos más alejados del mismo, opuestos a la estimación prevista. Por este motivo, los gráficos dificultan la comprensión de los resultados obtenidos en la tabla, mostrando soluciones aparentemente contrarias a los argumentos previos.

Parece demostrarse la sensibilidad al exponente como medida de penalización para desviaciones extremas. Sin embargo, no existen diferencias significativas a la hora de emplear un exponente u otro como parámetro de distorsión, por lo que resulta difícil determinar su contribución en el sistema.

5.3.5.2. RESULTADOS CON DIFERENTES RADIOS Y NÚMEROS DE OBJETIVOS

Objetivos	Medidas			Valores	
2	Media			15,11	
	Mediana			15,17	
	Desviación típica			1,67	
Objetivos	Rango	Exponente	Estabilidad	Medidas	Valores
3	5%	1	Alta	Media	12,86
				Mediana	12,76
				Desviación típica	0,60
			Media	Media	17,43
				Mediana	17,44
				Desviación típica	0,92
			Baja	Media	19,80
				Mediana	20,28
				Desviación típica	2,36
3	10%	1	Alta	Media	14,35
				Mediana	14,41
				Desviación típica	0,64
			Media	Media	19,08
				Mediana	19,01
				Desviación típica	0,95
			Baja	Media	22,51
				Mediana	22,65
				Desviación típica	1,41
3	15%	1	Alta	Media	15,07
				Mediana	15,02
				Desviación típica	0,71
			Media	Media	20,13
				Mediana	19,94
				Desviación típica	1,09
			Baja	Media	22,12
				Mediana	22,48
				Desviación típica	3,54

Tabla 29: análisis de la robustez con el radio

La tabla refleja resultados exactamente contrarios a los esperados. A medida que se incrementa el rango de variación en las distorsiones de la cartera, empeora la estabilidad, produciéndose alteraciones más grandes ante los errores de estimación. De hecho, los frentes obtenidos con un radio del 15 % son muy similares a los obtenidos sin tener en cuenta el objetivo de la estabilidad.

Análisis de la robustez con el radio										
	BaseAlta	BaseBaja	BaseMed	2Obj	2xRALta	2xRBaja	2xRMed	3xRALta	3xRBaja	3xRMed
BaseAlta										
BaseBaja	++									
BaseMed	++	--								
2 Obj	++	--	--							
2xRALta	++	--	--	--						
2xRBaja	++	++	++	++	++					
2xRMed	++	--	++	++	++	--				
3xRALta	++	--	--	=	++	--	--			
3xRBaja	++	++	++	++	++	=	++	++		
3xRMed	++	=	++	++	++	--	++	++	--	

Tabla 30: Comparativa análisis de la robustez con el radio

Los contrastes estadísticos confirman los resultados mostrados en la tabla 28. Todas las comparaciones realizadas con el caso base de estabilidad alta, demuestran ser peores que ella ante errores de estimación, con un nivel significativo del 1%. Los rangos de variación del 10% y 15%, solo consiguen menores dispersiones en los resultados obtenidos, con respecto a los niveles de mayor inestabilidad. Las estimaciones en los frentes de Pareto con dos objetivos son incluso más exactas que las obtenidas con dos objetivos. Por lo tanto, a la vista de estos resultados, el sistema no parece ser sensible a variaciones producidas con un rango mayor del 0.05%.

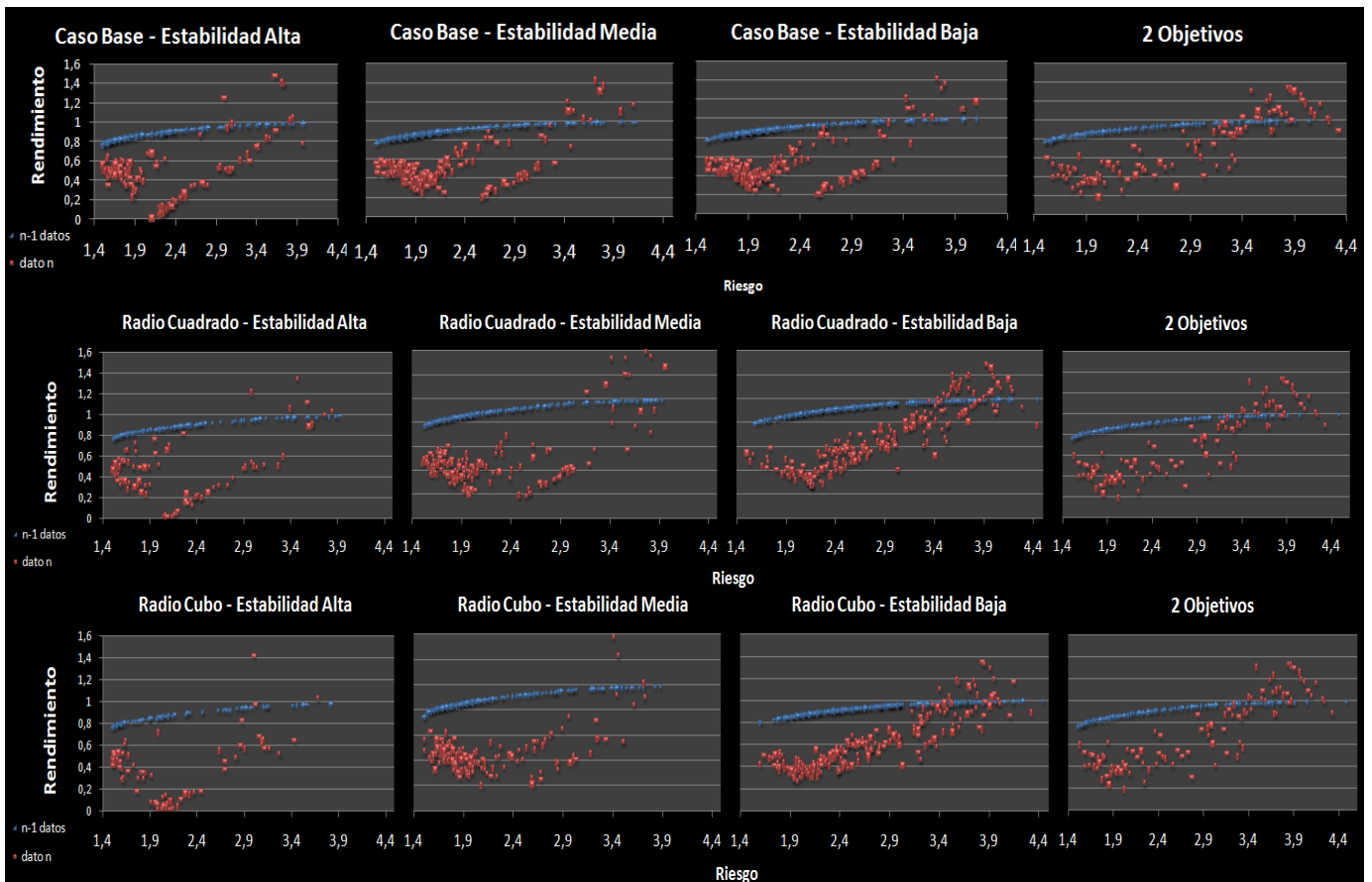


Figura 41: Análisis gráfico resultados experimento con distintos rangos de variación

Pueden observarse unos resultados muy parecidos a los que se obtuvieron en el análisis del exponente. Las gráficas vuelven a mostrar resultados aparentemente distintos a los mostrados en la tabla 28. La concentración de puntos parece más próxima a los frentes de Pareto estimados, cuanto menor es la estabilidad, produciéndose resultados muy dispersos para las estabildades más altas. Las distribuciones de puntos son muy similares entre sí, en función de la estabilidad, con independencia del rango de variación introducido en la cartera original.

A continuación se analizarán las distancias de Mahalanobis resultantes al ejecutar la aplicación utilizando remuestreo, para poder comprobar la influencia que esta medida de distorsión ejerce sobre los resultados finales. En el análisis anterior, llevado a cabo entre estas dos configuraciones, no pudo obtenerse ninguna conclusión, al no poder compararse los resultados con el mismo frente de referencia, por tratarse de escenarios completamente diferentes. Con esta nueva métrica se pretende poder llevar a cabo una comparativa real de la aportación de nuevos escenarios como medida de distorsión.

5.3.5.3. RESULTADOS CON REMUESTREO

Objetivos	Medidas		Valores	
2	Media		15,11	
	Mediana		15,17	
	Desviación típica		1,67	
Objetivos	Remuestreo	Estabilidad	Medidas	Valores
3	SI	Alta	Media	13,22
			Mediana	13,21
			Desviación típica	5,69
		Media	Media	13,74
			Mediana	13,69
			Desviación típica	2,18
		Baja	Media	15,58
			Mediana	15,54
			Desviación típica	5,39
3	NO	Alta	Media	12,86
			Mediana	12,76
			Desviación típica	0,60
		Media	Media	17,43
			Mediana	17,44
			Desviación típica	0,92
		Baja	Media	19,75
			Mediana	20,01
			Desviación típica	2,15

Tabla 31: Resultados con remuestreo.

Las distancias de Mahalanobis más pequeñas se obtienen para los niveles más estables. Los resultados obtenidos al introducir remuestreo como parámetro de distorsión, son muy similares a los del caso base, llegando a ser mejores para un nivel de estabilidad media. A continuación analizaremos si estas distancias son o no significativas.

Análisis remuestreo vs. caso base							
	AleatorioAlta	AleatorioBaja	AleatorioMedia	C.BaseAlta	C.BaseBaja	C.BaseMedia	2Objetivos
AleatorioAlta							
AleatorioBaja	++						
AleatorioMedia	=	--					
C.BaseAlta	=	--	--				
C.BaseBaja	++	++	++	++			
C.BaseMedia	++	++	++	++	--		
2Objetivos	++	=	++	++	--	--	

Tabla 32: Análisis estadístico remuestreo

En la tabla de contrastes puede apreciarse que no existen diferencias significativas cuando se trabaja con el caso base o con remuestreo (o aleatorio). Para el resto de los niveles de estabilidad, las soluciones obtenidas a partir de remuestreo son más robustas que los frentes del caso base, con una diferencia significativa del 1%. Su distancia de Mahalanobis son más pequeñas a los frentes estimados.

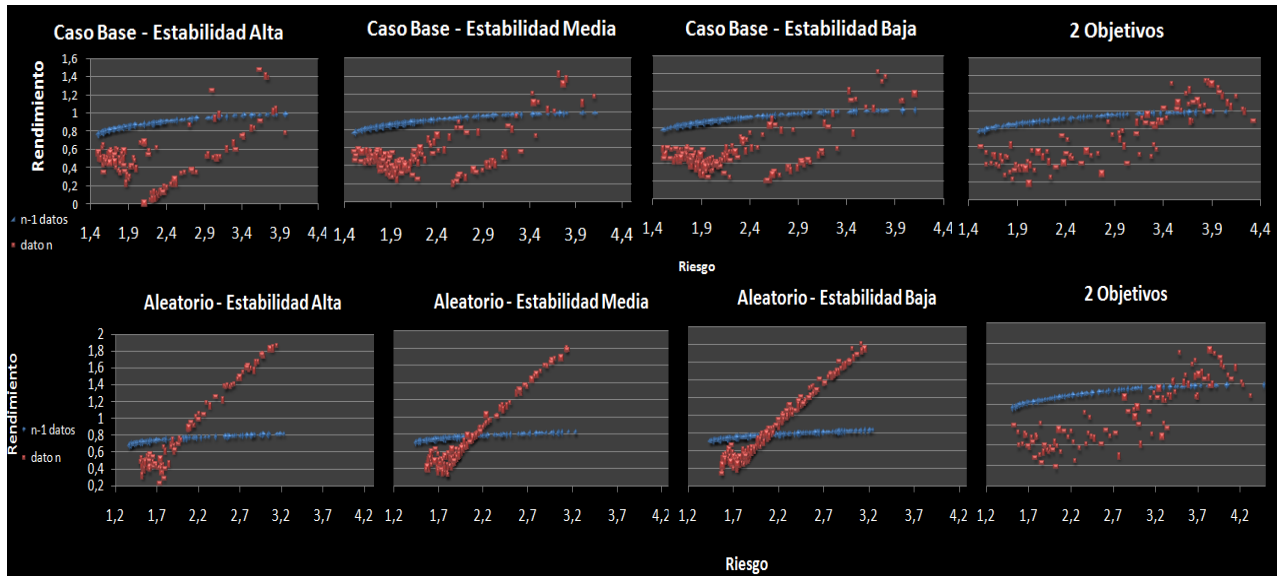


Figura 42: Análisis gráfico resultados experimento con remuestreo

En las gráficas se muestran soluciones muy distintas según se utilice remuestreo o no. Las soluciones obtenidas al introducir remuestreo en el sistema, tienen forma similar a un frente de Pareto, completamente diferente al azul que se había estimado. Para el caso base, ocurre lo mismo que en el resto de las gráficas descritas anteriormente: conforme disminuye la estabilidad, los puntos parecen situarse más próximos a los frentes estimados. Pese a que los gráficos parecen indicar lo contrario, esta distorsión ha demostrado ser una buena medida de estabilidad. No se producen diferencias significativas en los errores de estimación entre el caso base y el remuestreo, para un nivel de estabilidad alta.

Capítulo 6

Conclusiones

6. CONCLUSIONES

En este capítulo se recogen las conclusiones extraídas de la realización de este Proyecto Fin de Carrera. Dentro de estas conclusiones se hace balance de las aportaciones que ofrece este proyecto, repasando en qué medida se han visto satisfechos los objetivos propuestos. Además, se resumirán las conclusiones extraídas de la experimentación para explicar sucintamente si se ha conseguido un sistema de optimización de carteras robusto y estable a lo largo del tiempo, comentando las ventajas e inconvenientes de la aplicación de algoritmos genéticos multiobjetivo como técnica de resolución para este tipo de problemas. Finalmente, se perfilan algunos de los trabajos futuros que pueden desarrollarse a partir del mismo, así como los contratiempos a los que se han tenido que hacer frente durante su desarrollo, para concluir con una visión personal acerca de todo el proceso.

6.1 OBJETIVOS ALCANZADOS

El proyecto desarrollado cumple con todos los objetivos que se habían establecido inicialmente:

- Se ha definido, diseñado e implementado un sistema de optimización de carteras de inversión multiobjetivo. El sistema es capaz de identificar las carteras que maximizan la rentabilidad, minimizando el riesgo y que aporta robustez ante errores de estimación de los parámetros del modelo
- El sistema tiene como base el algoritmo genético de optimización multiobjetivo SPEA2 y se ha implementado de tal forma que puede operar bajo restricciones de cardinalidad y límites máximos y mínimos de inversión en cada activo.
- Se ha realizado un análisis que ha permitido discriminar qué formas de modelar la robustez entre las propuestas resultan efectivas. Así mismo, se ha estudiado la relación entre los valores de sus parámetros y la calidad de las soluciones

6.2 CONCLUSIONES FINALES:

El problema de optimización de carteras es un escenario natural para el uso de algoritmos genéticos multiobjetivo, sobre todo por la capacidad y flexibilidad que presentan estos tipos de algoritmos. En este sentido, este trabajo ha utilizado el algoritmo dos diferentes estados del arte en esta área específica, es decir, NSGAI y SPEA2, utilizando como marco experimental valores reales de la bolsa estadounidense. Tras una larga fase de experimentación y análisis, se puede afirmar que la introducción de la estabilidad como tercer objetivo, permite la obtención de soluciones más próximas a la realidad que trabajando solo con los dos objetivos del Modelo de Media-Varianza de Markowitz.

La comparación inicial de estos algoritmos (NSGAI y SPEA2), tenía como objetivo confirmar que SPEA2 es una buena opción para abordar el problema de optimización de carteras de inversión. Para validar el desempeño de cada uno ellos, se llevaron a cabo 42

ejecuciones independientes, cuyos frentes resultantes se compararon con un frente de referencia, obtenido a partir de los individuos no dominados de entre un total de 8400 (84 ejecuciones con 100 individuos cada una). El rendimiento de cada algoritmo se evaluó mediante contrastes estadísticos realizados a partir de los resultados obtenidos en las métricas de hipervolumen y spread. De esta forma pudo demostrarse que el algoritmo de SPEA2 se comportaba mejor para este dominio. Esto apoya la afirmación de que se trata de uno de los mejores algoritmos genéticos multiobjetivo.

Una vez realizada esta prueba inicial, el resto de los experimentos se centraban en evaluar la sensibilidad a las modificaciones de los parámetros de distorsión introducidos, para simular errores de estimación en los rendimientos y riesgos esperados. En nuestro intento de construir un sistema capaz de realizar predicciones lo más ajustadas a la realidad, tolerante a pequeñas equivocaciones, es importante analizar ante qué parámetros se consigue una mayor tolerancia y, por consiguiente, una mayor estabilidad.

Los parámetros de distorsión demostraron ser sensibles, tanto al radio, como al exponente. A medida que se introducía mayor variabilidad en las carteras originales, se obtenían soluciones más robustas. De igual modo, cuando se penalizaban las distancias entre estas variaciones elevándolas al cuadrado y exponente, se alcanzaban soluciones más estables. Sin embargo, no pudo demostrarse qué exponente era mejor, al obtener resultados sin ninguna diferencia significativa.

La evaluación inicial del remuestreo resultó de una interpretación más difícil de la prevista, ya que la estructura de los frentes obtenidos era bastante diferente a las del caso base. En todo caso, se vio que llevaba a frentes muy distintos a los que se obtenían sin aplicarlo. La calidad en términos de hipervolumen bajaba en un grado importante.

Tras la conclusión de este análisis quedaba demostrar hasta qué punto las dos técnicas propuestas para obtener frentes robustos, el tercer objetivo y el remuestreo, resultaban ser de utilidad. Para ello, se midieron las rentabilidades y riesgos de la cartera de las desviaciones promedio entre los frentes estimados obtenidos con los parámetros de distorsión, y el comportamiento de estas mismas carteras con los valores observados. A continuación, se comparan las distancias de Mahalanobis promedio entre las rentabilidades y riesgos previstos de las carteras incluidas frentes estimados, y las observadas.

Los resultados obtenidos tras los contrastes estadísticos, demostraron que la técnica propuesta obtuvo frentes de Pareto más cercanos a los datos reales, que trabajando solo con dos objetivos. Las distancias desde los puntos del frente previsto al efectivo resultaron ser menores, por lo que las soluciones obtenidas al incluir el objetivo de la estabilidad son más robustas a los cambios de los valores de los parámetros. También pudo validarse la aportación del exponente como medida de penalización para medidas extremas, demostrándose que resultados óptimos, con menores errores en la estimación, se alcanzaron al elevar al cubo el exponente. El rango de variación introducido para las variaciones de la cartera original, resultó ser mejor cuanto menor era su valor, obteniéndose valores poco robustos a medida que se introducía una mayor dispersión. El empleo del remuestreo demostró ser otro parámetro de distorsión útil para evitar la hiperespecialización de un escenario particular.

En definitiva:

- a) Los algoritmos genéticos multiobjetivo son capaces de resolver el problema de la optimización de carteras con restricciones de cardinalidad y límites de inversión.
- b) Según los resultados obtenidos, SPEA2 resulta superior a NSGA-II para resolver el problema básico propuesto por Markowitz. Esto es consistente con los resultados de otros estudios.
- c) Tanto el tercer objetivo de robustez propuesto, como el remuestreo rebajan la calidad del frente en términos de rentabilidad y riesgo calculado usando los parámetros estimados
- d) El tercer objetivo y el remuestreo mejoran la robustez de las carteras estimadas.
- e) Los cambios en el exponente del tercer objetivo afectan de forma variable a la calidad de los frentes estimados y a su robustez ante errores de estimación de parámetros
- f) Los cambios en el radio del tercer objetivo afectan de forma variable a la calidad de los frentes estimados e inversa su robustez ante errores de estimación de parámetros.

6.3 LÍNEAS FUTURAS DE INVESTIGACIÓN

Como trabajo de investigación a partir de este proyecto, se propone:

- a) Analizar la estabilidad del sistema para otros escenarios de tiempo distintos al que se ha evaluado. Este proyecto se ha centrado en intentar proporcionar robustez al modelo de Markowitz, enfocando el análisis de la aportación de la estabilidad sobre el último dato de las series de rendimientos mensuales disponibles. Conocidos los datos anteriores, se ha demostrado obtener mejores aproximaciones a los valores de rendimiento y riesgo previstos mediante la función de robustez, que si no se tiene en cuenta esta función. Sin embargo, sería conveniente realizar una mayor cantidad de pruebas con distintos escenarios de tiempo, para medir hasta qué punto son generalizables los resultados obtenidos.
- b) Realizar una mayor comparativa de los parámetros disponibles (incluyendo los valores de las restricciones), aplicando otras combinaciones de los mismos, y/o modificando su valor.
- c) Evaluar la eficiencia computacional. Para facilitar su cálculo, ya se mide el tiempo de cada ejecución, mostrado por pantalla al final de las mismas.
- d) Evaluar nuevas formas de caracterizar la robustez.
- e) Analizar los resultados no sólo midiendo la distancia entre los resultados posibles y los obtenidos, sino también entre los obtenidos y los mejores posibles para ese período.
- f) Debido al enfoque práctico de este proyecto, sería relevante divulgar este tipo de soluciones a los interesados directos en la solución del modelo de Media-Varianza, a

la vez que determinar otro tipo de problemas y características reales del ambiente económico-financiero, con la finalidad de implementar su solución mediante la aplicación de los algoritmos genéticos multiobjetivo.

6.4 PROBLEMAS ENCONTRADOS

El desarrollo de este proyecto no ha estado exento de dificultades, las cuales se han localizado básicamente en el desconocimiento inicial del dominio del problema, en el aprendizaje del framework utilizado, en la verificación de los resultados obtenidos, y en la reducción del tiempo disponible para su desarrollo.

Antes de empezar con este proyecto tenía un total desconocimiento sobre qué eran las carteras de inversión, cómo se trabajaba con ellas, y toda la terminología derivada del mundo de las finanzas, así como las restricciones y dificultades que implicaba su optimización. Tampoco había trabajado nunca con algoritmos genéticos multiobjetivo, por lo que también desconocía sus aplicaciones, alcance y modo de empleo. Fue necesaria una fase de documentación como preparación previa al desarrollo del proyecto.

Nunca antes había trabajado con un dominio parecido ni utilizado ningún framework de estas características, por lo que no conocía qué me podía aportar en el desarrollo del mismo ni cómo se manejaba. Por este motivo, he tenido que emplear bastante tiempo, más del inicialmente planeado, en el estudio y aprendizaje tanto del dominio del problema, como de las tecnologías mencionadas, incluyendo el manejo del framework. Pero este tiempo se ha visto compensado con el esfuerzo de la implementación, pues conocidos los objetivos, dónde y cómo implementarlos en el framework y la aportación de los mismos, su codificación se ha desarrollado con cierta facilidad. Otro de los problemas tecnológicos al que he tenido que hacer frente ha sido la comprensión del manejo del paquete *experiments* de JMetal para poder realizar los experimentos de la fase de Experimentación, de la forma más dinámica posible. Al necesitarse un gran número de ejecuciones para la obtención de métricas de desempeño que permitiesen evaluar el rendimiento de los algoritmos, cualquier ayuda que permitiese su automatización sería bien recibida, y en las clases de este paquete se conseguía nuestro objetivo (tras adaptarlas a nuestro problema). Esto, a su vez, implicaba manejar *Latex* y programas estadísticos como *R*, a un nivel básico, para poder analizar los resultados, cuya verificación ha supuesto el otro gran problema mencionado al que me he tenido que enfrentar.

Al tratarse de algoritmos genéticos que parten de una población aleatoria, distinta entre diferentes ejecuciones, el proceso de validación de los resultados ha sido bastante complejo. Por ello, tanto su implementación como verificación posterior, exigían una forma de trabajar muy metódica, de modo que el código requería independencia en cuanto a los objetivos, cuya implementación se llevó a cabo de forma incremental en términos de dificultad. En primer lugar se implementó un ejemplo sencillo, extraído del libro utilizado en la documentación. Consistía en la optimización de carteras con los dos objetivos del Modelo de Markowitz. De este modo pudieron validarse uno de ellos, el de la rentabilidad, y ciertas características del objetivo de riesgo: la matriz de varianzas-covarianzas. Sin embargo, el resultado final del objetivo de riesgo tuvo que validarse una vez implementada la aplicación, con un ejemplo proporcionado por mi tutor, dado un individuo en particular como población inicial. A continuación, era necesario comprobar que los algoritmos genéticos multiobjetivo

funcionaban perfectamente, y que los objetivos se maximizaban y minimizaban, según correspondiese, de forma correcta. Para ello se utilizaron varios programas de optimización de carteras, implementados con algoritmos exactos en Excel, capaces de resolver el problema relajado, sin restricciones. Verificar el cumplimiento de las restricciones era más sencillo, bastaba con analizar trazas de la ejecución y comprobar que todas ellas se cumplieran. Finalmente, el tercero de los objetivos se validó con un código escrito en *Matlab*, facilitado también por mi tutor, donde podía verificarse que el valor de la distancia de Mahalanobis coincidía con el de nuestra aplicación.

Pero la fase de experimentación fue, sin duda, la más compleja y laboriosa. El planteamiento de las pruebas iniciales para la selección del mejor algoritmo genético, requería la obtención de un frente de referencia óptimo con el que poder realizar las comparaciones para la obtención de las métricas. Este mismo frente se empleó también para el análisis de parámetros de distorsión del tercer objetivo. Para poder llevarlo a cabo hubo que centrar la tención solo en los dos primeros objetivos, y obtener las soluciones no dominadas de los mismos, lo que supuso nuevas modificaciones. A continuación, se planteó la posibilidad de dividir estos resultados por niveles de estabilidad, para un mejor análisis de los parámetros de la distorsión y de la aportación de la robustez en el sistema. Finalmente, para evaluar la proximidad de los resultados reales con respecto a los frentes previstos, fue necesaria la implementación de una nueva métrica, que midiera las distancias de Mahalanobis entre ellos. A partir de estos resultados, se llevaron a cabo, de igual modo que para las métricas de spread e hipervolumen, una serie de contrastes estadísticos, por medio de un script que se implementó en R, al que también hubo que dedicar cierto tiempo.

El otro gran problema al que he tenido que hacer frente ha sido el tiempo. Desde un principio se ha planteado una planificación muy optimista, contando con unos cuatro meses (ampliables) para la realización total del proyecto, incluyendo tanto el desarrollo de la aplicación como la redacción de la memoria. Sin embargo, durante el transcurso de este período, uno de mis tutores me informó de su intención de viajar a Estados Unidos. Puesto que me gustaría que estuviera presente durante la defensa del proyecto, he tenido que dedicar un gran esfuerzo para tratar de cumplir con los plazos previstos y que esto no afectara a la calidad ni de la aplicación ni de la memoria.

6.5 OPINIONES PERSONALES

El desarrollo de este proyecto me ha supuesto un gran esfuerzo y dedicación para conseguir acabarlo a tiempo y aún así mantener la calidad que un proyecto de estas características precisa. Su realización me ha aportado conocimientos de otras técnicas de algoritmos genéticos que no conocía, como son los algoritmos genéticos multiobjetivo, y de herramientas con las que poder utilizarlos. Me ha ofrecido la posibilidad de trabajar en dominios totalmente nuevos para mí, permitiéndome corroborar el alcance y el impacto de la Informática en multitud de campos diferentes, como en el área de las Finanzas para este estudio en concreto, sin necesidad de ser un experto en el tema. También me ha proporcionado conocimientos en el desarrollo de proyectos (especialmente en la gestión del tiempo) y, sobre todo, en el trabajo a nivel individual.

Hasta ahora, durante la carrera, la mayor parte de las prácticas que he tenido que realizar han sido siempre en parejas, y si ésta suponía un gran esfuerzo, se realizaban en

grupo. El Proyecto de Fin de Carrera, sin embargo, se desarrolla individualmente, sin depender de otros compañeros que te puedan ayudar y a los que rendir cuentas, lo que me ha permitido aprender a superar los problemas, tomar decisiones y planificar el tiempo por mí misma.

En cuanto al tema del tiempo, como ya se ha explicado en la sección anterior, me ha supuesto un reto al intentar ajustarme a la fecha de entrega, tras haberme planificado inicialmente contando con un mayor plazo. Sin embargo, gracias a ello, he aprendido a adaptarme a nuevas situaciones y organizar mejor el tiempo disponible, planificar con anterioridad las tareas a desarrollar y cumplir los nuevos plazos establecidos con independencia de las horas diarias que tuviera que dedicarle al proyecto.

El estudio, comprensión y posterior análisis de los algoritmos genéticos multiobjetivo para su aplicación en la optimización de carteras de inversión, me ha supuesto invertir bastante tiempo y dedicación. Sin embargo, me ha permitido adquirir conocimientos sobre otra técnica de la Inteligencia Artificial, con numerosas ventajas en cuanto a velocidad, recursos computacionales y exactitud se refiere, frente a otras técnicas más tradicionales (basadas en heurísticas o exactas), y con un gran número de aplicaciones en áreas de la más diversa índole. Lo mismo ha ocurrido con la utilización del framework. Al principio, supuso bastante tiempo aprender todas las funcionalidades que proporcionaba pero, finalmente, ha merecido enormemente la pena su uso, y conforme más avanzaba en la implementación, más funcionalidades nuevas descubría que me facilitaban el trabajo.

La elaboración de la memoria me ha permitido ampliar mis conocimientos sobre la documentación de proyectos y la importancia de que ésta esté explicada con claridad. También he aprendido a investigar, analizar los resultados de las métricas de desempeño y frentes de Pareto y documentar con la bibliografía adecuada.

En resumen, la elaboración de este proyecto ha supuesto un gran esfuerzo y dedicación, pero también me ha permitido adquirir numerosos conocimientos muy útiles para el desarrollo de futuros proyectos y trabajos de investigación.

Anexo I

Seguimiento de Proyecto

ANEXO I: SEGUIMIENTO DE PROYECTO DE FIN DE CARRERA

Debido a la necesidad de tener que presentar este proyecto en septiembre de 2010 y no poder demorar esta fecha bajo ningún concepto, ha resultado indispensable fijar una planificación adecuada, que se adaptara a las necesidades del proyecto y tratar de seguirla con la mayor fidelidad posible. En esta sección se presentan tanto la planificación inicial como la real, y se hace una comparativa entre ambas.

PLANIFICACIÓN INICIAL

A continuación se muestra la planificación que se estimó al comienzo del proyecto, contando con que la fecha de entrega del proyecto sería a mediados de septiembre, y que esta fecha como mucho podría posponerse una o dos semanas. Se puede ver que para la fase de desarrollo de la aplicación se calculó inicialmente una dedicación de dos meses, sin embargo, esta dedicación es parcial, ya que durante el mismo tiempo se estimó que se podría ir avanzando la memoria. La planificación inicial se recoge en la figura 42.

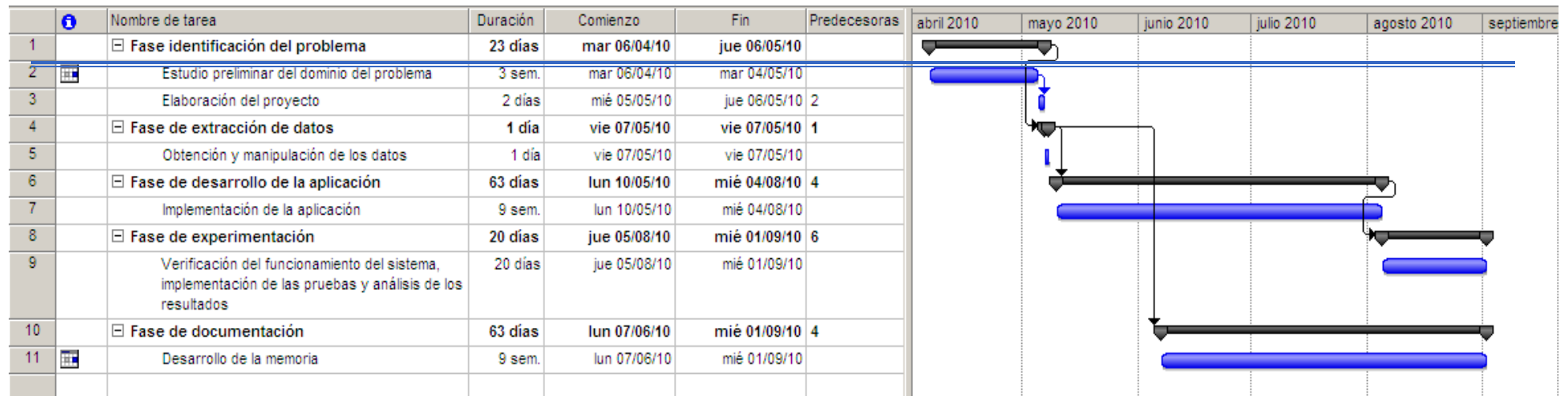


Figura 43: Planificación inicial

PLANIFICACIÓN FINAL

Aunque se ha intentado cumplir con la planificación inicial, no siempre ha sido posible, como consecuencia de ciertas tareas que han llevado más tiempo del inicialmente previsto. Tal es el caso del tiempo estimado para el desarrollo de la fase de “Experimentación”, que ha supuesto una dedicación mayor de la planificada, debido, principalmente, a la gran cantidad de pruebas llevadas a cabo y al posterior análisis de las mismas. Como consecuencia de este retraso, el desarrollo de la memoria también se vio afectado. La implementación del código tampoco se libró de un cierto desfase con respecto al tiempo estimado, debido, especialmente al sobreesfuerzo dedicado a la validación de los resultados. A pesar de estos contratiempos, se ha logrado mantener la fecha de entrega (aunque un par de semanas más tarde de lo que se hubiera deseado).

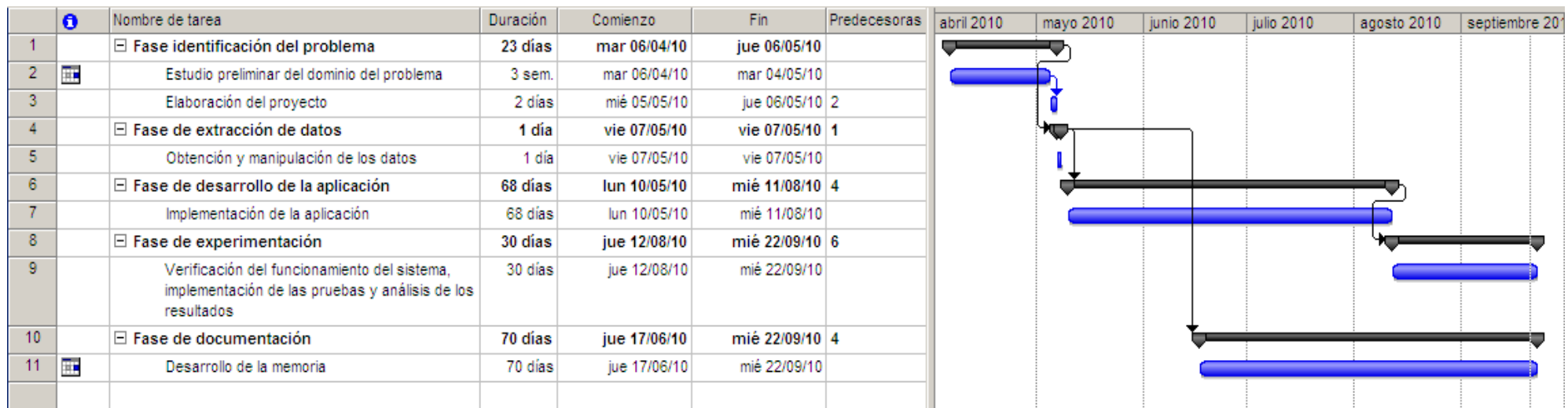


Figura 44: Planificación final

PRESUPUESTO INICIAL

En la siguiente tabla se recoge el presupuesto inicial que se estimaba que costaría la aplicación. Sin embargo, al emplearse más tiempo en su construcción del inicialmente propuesto, debe contemplarse un presupuesto final que contemple esta diferencia de tiempo.

Descripción	Importe
Gastos de personal	23.350 €
Gastos hardware	166,68 €
Gastos software	1.507,15 €
Gastos derivados	232,80 €
Total costes directos	25.256,63 €
Riesgos (10%)	2.525,66 €
Beneficios (10%)	2.525,66 €
Total sin I.V.A	30.307,95 €
IVA (18%)	54.55,43 €
Total a pagar	35.763,38 €

Tabla 33: Presupuesto inicial

PRESUPUESTO FINAL

Los gastos finales del proyecto fueron los mismos en los conceptos de hardware, software y otros gastos. Pero, el retraso en la entrega final del proyecto ha hecho aumentar el gasto de personal. En concreto, se ha retrasado 15 días, aumentando la cuantía de horas en 15 días laborables * 9 horas/día = 135 horas. Por lo que se produce un aumento de 2.700€ en gastos de dicha partida. Inicialmente se estimó un precio de 30.307,95€ y, finalmente, se ha gastado 28.006,63€, por lo que se obtendría un beneficio de 2.301,32€. Como vemos, el retraso ha afectado al margen de beneficios inicial, si bien el haber contado con un colchón de riesgo nos permite finalizar el proyecto con beneficios a pesar del retraso.

Descripción	Importe
Gastos de personal	26.050 €
Gastos hardware	166,68 €
Gastos software	1.507,15 €
Gastos derivados	232,80 €
Total costes directos	28.006,63 €

Tabla 34: Gastos finales

Anexo II

Manual de usuario

ANEXO II: MANUAL DE USUARIO

En esta sección se recoge el manual de usuario que acompaña a la aplicación desarrollada y que sirve como guía para su manejo.

¿PARA QUÉ SIRVE LA APLICACIÓN?

Esta aplicación está diseñada con el objetivo de optimizar carteras de inversión robustas, mediante el empleo de algoritmos genéticos multiobjetivo. Además, permite realizar distintos tipos de experimentos sobre los frentes de Pareto resultantes, y ejecutar la aplicación con dos y tres objetivos. Facilita la modificación de los parámetros de configuración y de entrada, permitiendo varias alternativas para su configuración, a fin de proporcionar una mayor flexibilidad para el manejo de la aplicación. El sistema muestra un menú con diferentes posibilidades de estudio de la estabilidad de una cartera de inversión. Se trata de un menú sencillo e intuitivo que guiará al usuario durante la ejecución del programa, proporcionándole distintos tipos de experimentos a realizar.

En este manual se explica cómo utilizar la aplicación desarrollada en este proyecto paso a paso, permitiendo al usuario realizar experimentos con las distintas opciones de configuración para los algoritmos propuestos.

Con este propósito se explican los siguientes aspectos:

- Requisitos mínimos.
- Formato de los ficheros de datos.
- Ejecución de la aplicación.
- Datos de salida.
- Ejemplo sencillo de ejecución de la aplicación.

REQUISITOS MÍNIMOS:

- Para ejecutar la aplicación se requiere tener instalado el entorno de ejecución java (JRE, *Java Runtime Enviroment*) 1.6 o superior⁴. Se puede ejecutar la aplicación en cualquier sistema compatible con dicho entorno de ejecución, independientemente del Sistema Operativo.
- Se recomienda un entorno de desarrollo gráfico como *Netbeans* o *Eclipse* para la ejecución de la aplicación.

⁴ Se puede obtener en <http://java.sun.com>

- Para realizar los contrastes estadísticos de las métricas de desempeño, se requiere tener instalado el entorno de programación R⁵. Puede ejecutarse sobre cualquier Sistema Operativo, y no es necesario conocer este lenguaje para poder realizar los contrastes, ya que se proporciona el script necesario.

FORMATO DEL FICHERO DE ENTRADA

Se utiliza un fichero de texto plano, en el que cada parámetro se representa en una línea diferente. En la figura 44, se representa un fichero con formato válido. Para que el formato de los ficheros sea correcto deben cumplirse las siguientes condiciones:

- En primer lugar debe especificarse el modo de ejecución: con remuestreo (aleatorio) o sin remuestreo (no aleatorio). El modo en que debe escribirse este parámetro es: “aleatorio” (para la ejecución con remuestreo) o cualquier otra palabra para la ejecución normal (sin remuestreo, utilizando todos los datos de las series de rendimientos mensuales que componen los activos a optimizar en orden cronológico).
- El segundo de los parámetros se corresponde con la restricción de cardinalidad, y hace referencia al número de activos distintos que pueden incluirse en la cartera de inversión. Este número es de tipo entero y, lógicamente, tiene que ser menor o igual al número de activos totales que componen la cartera.
- Los parámetros tercero y cuarto hacen referencia a los umbrales de compra máximos y mínimos, de modo que la tercera línea del fichero se corresponde con el umbral máximo de cada uno de los activos, y la cuarta con el umbral de compra mínimo. Deberán escribirse tantos números reales como activos compongan la cartera, separados entre sí mediante un tabulador. Cada uno de ellos será representable según el estándar IEEE 754. Utilizándose el punto como separador decimal. Hay que tener en cuenta que los números de la tercera línea deberán ser mayores o iguales a los de la siguiente línea, y ambos estarán comprendidos entre los valores 0.0 y 1.0.
- El quinto parámetro expresa la totalidad del tiempo, medido en meses, que componen las series de rendimientos mensuales con la que se está trabajando. Se trata de un número entero.
- El sexto parámetro se corresponde también con un número entero, que representa la cantidad de activos que componen la cartera de inversión.
- El séptimo parámetro indica el número de carteras de inversión creadas a similitud de la que se está evaluando. Se utiliza para obtener el valor promedio de la distancia de Mahalanobis, encargado de medir la variación estimada en torno a las modificaciones reales que podrían producirse en el mercado.
- Los parámetros octavo y noveno expresan el rango de variación explicado en el punto anterior. La línea octava se corresponderá con el límite superior del rango, mientras que la siguiente línea hace referencia al límite inferior de dicho rango, y por

⁵ Se puede descargar este intérprete en <http://cran.r-project.org/>

tanto se expresa con signo negativo. Ambos límites serán números reales próximos a cero.

- A continuación, se encuentran las series de datos históricos con los que se va a trabajar para optimizar las carteras. Cada fila se corresponde con los activos que componen las carteras de inversión, y cada columna con los rendimientos durante un período de tiempo para ese activo. Las carteras o son números reales, positivos o negativos, separados unos de otros mediante un tabulador.
- Cada uno de los atributos con valores numéricos reales será representable según el estándar IEE 754. Se utiliza el punto como separador decimal.
- Si un parámetro se compone de más de un valor, éstos se encuentran separados por tabuladores.
- Los distintos parámetros descritos anteriormente se encuentran separados por saltos de línea. Cada línea hace referencia a un parámetro diferente, salvo para las series de rendimientos mensuales, donde cada línea representa una cartera de inversiones diferente.
- El tamaño de los ficheros queda limitado por el hardware del sistema, ya que los datos se cargan en memoria para poder realizar los cálculos una vez leídos y procesados.

Noaleatorio							
4							
0.6	0.7	0.8	0.9	1.0	0.6	0.7	0.65
0.05	0.03	0.2	0.0	0.1	0.05	0.3	0.15
6							
326							
25							
-0.1							
0.1							
1							
6.795972226	9.597079548	4.594094043	7.231728672	6.024236775	4.390097855		
-3.943245668	-0.369032274	-0.917318381	-0.100272167	-0.924552149	1.568372907		
-16.10677909	-20.67704365	-10.79699173	-12.30857471	-13.97689929	-7.190203014		
5.875733127	6.207191277	5.108656334	4.17293479	3.70573349	3.378010505		
7.931366098	8.026844746	5.63070814	4.39819175	6.080387951	0.851953685		
2.9955814	4.982999299	2.100145356	4.309126447	5.629304029	3.944246055		
9.054207877	12.96314983	4.619538537	8.120964022	4.74809965	1.275941811		
4.678932279	8.090087079	0.586551546	1.607785629	1.608951864	1.38023607		
0.942175424	4.936046791	0.76506811	4.398026661	5.588426952	1.96373318		

Figura 45: Ejemplo de fichero de datos válido

PÁRAMETROS DE CONFIGURACIÓN:

Todos los parámetros de la aplicación se configuran en un fichero de configuración, o interactuando con el menú, a través de la entrada estándar, por teclado. Estos parámetros se corresponden con la ubicación del archivo correspondiente al experimento a ejecutar, la ruta donde se almacenan los frentes de referencia (por defecto relativa), el número máximo de generaciones, el tamaño de la población inicial y las características de los algoritmos genéticos: tipos de operadores de cruce, mutación y selección empleados, así como las

probabilidades de que los individuos muten o se crucen en cada generación, junto con el índice de distribución. En la tabla anexo3 se detallan las propiedades generales de configuración, especificando los valores que cada parámetro de configuración puede tomar:

Campo	Descripción	Valores
Ruta del experimento	Se especifica la ubicación del experimento que se va a ejecutar.	Por defecto, se almacena en la misma carpeta donde se tiene el proyecto.
Ruta de los frentes de referencia	Debe especificarse la ubicación de los frentes de referencia con los que se compararán los frentes generados para calcular las métricas.	Por defecto, se almacena en la misma carpeta donde se tiene el proyecto.
populationSize	Se especifica el tamaño de la población de individuos. Esta cifra coincidirá con el número de valores obtenido en el frente de Pareto resultante de la ejecución de la aplicación.	Admite números naturales mayores que dos. Es preferible tamaños de poblaciones grandes, de más de 50 individuos.
archivoSize	Tamaño del archivo externo utilizado por SPEA2 para almacenar las soluciones no dominadas obtenidas en cada generación.	Admite número naturales mayores que dos. Debe utilizarse el mismo tamaño que el de la población.
maxEvaluations	Criterio de parada del algoritmo. Establece el número máximo de iteraciones que realizará el algoritmo para alcanzar la solución y devolver el frente de Pareto.	Admite números enteros positivos. Se recomienda el empleo de números elevados, más de 10 veces superiores al tamaño de la población.
Cruce	Especifica la proporción de la población que se reproducirá en cada generación.	Las probabilidades de cruce admiten valores reales entre cero y uno. Por defecto: 0.9. Esta aplicación utiliza el operador de cruce: <i>SBXCrossover</i> .
Mutación	Especifica la proporción de la población que mutará en cada generación.	Las probabilidades de mutación admiten valores reales entre cero y uno. Por defecto: 1/número de variables de decisión. Esta aplicación utiliza el operador de mutación: <i>PolynomialMutation</i> .
Selección	La selección se realiza mediante torneo binario, utilizando un operador de comparación que permite seleccionar a los individuos con menor rango de dominancia. En el caso de empate se selecciona al individuo con mayor distancia de apilamiento (distancia media de dos puntos alrededor de la solución, a lo	Dependiendo del objetivo a conseguir y de las características del problema que se esté utilizando, puede elegirse un operador de entre un amplio conjunto implementado en el <i>framework</i> . Sin embargo, para este estudio, el operador que nos interesa es: <i>BinaryTournament</i> .

largo de todos los objetivos).

Tabla 35: Parámetros de configuración

A modo de ejemplo, a continuación se presenta un archivo de configuración válido para cualquier experimento a ejecutar. Si se decide modificar su contenido interactuando con el menú, los cambios producidos en este fichero respetarán este formato.

```
#Ruta del experimento actual a ejecutar
../EstudioEstabilidad/
#Ruta del frente de referencia
../FrentesDePareto/
#Tamaño de la poblacion
1000
#Tamaño del archivo
1000
#Numero de evaluaciones
20000
#Probabilidad de mutacion
1.0
#Probabilidad de cruce
0.9
#Indice de distribucion del cruce
20.0
#Indice de distribucion de la mutacion
20.0
#Numero de ejecuciones independientes
42
```

Figura 46: Ejemplo de parámetros de configuración

EJECUCIÓN DE LA APLICACIÓN

En la figura 46 se especifica cómo ejecutar la aplicación. No se requieren parámetros para su ejecución:

```
java -jar optimizacionCarteras.jar
```

Figura 47: Ejecución de la aplicación

Tras escribirse este comando, se mostrará un menú con los diferentes experimentos que se pueden ejecutar:

```
=====
                        OPTIMIZACIÓN DE CARTERAS DE INVERSIÓN
=====
Por favor, seleccione el experimento que desea ejecutar
1. Optimización de carteras robustas
2. Optimización de carteras con dos objetivos
3. Experimentos para el estudio de la estabilidad
4. Modificar parámetros del fichero de configuración
5. Salir
=====
1
```

Figura 48: Menú de la aplicación

Deberá escribirse el número correspondiente al experimento que se pretende ejecutar y pulsar *intro* para llevar a cabo su ejecución. El cometido de cada opción es el siguiente:

- **La opción 1** permite la obtención de frentes de Pareto teniendo en cuenta la estabilidad.
- **La opción 2** permite conseguir frentes de Pareto considerando solo el rendimiento a maximizar y el riesgo a minimizar.
- **La opción 3** permite trabajar con la estabilidad de los frentes de Pareto generados mediante la opción 1. Para su ejecución, se solicitará la ubicación de los frentes de Pareto generados tras la selección de la opción 1, aquéllos en cuyo tercer objetivo se centra este estudio:

```

=====
                        OPTIMIZACIÓN DE CARTERAS DE INVERSIÓN
=====
Por favor, seleccione el experimento que desea ejecutar
1. Optimización de carteras robustas
2. Optimización de carteras con dos objetivos
3. Experimentos para el estudio de la estabilidad
4. Modificar parámetros del fichero de configuración
5. Salir
=====
3
Introduzca la ruta de la carpeta con los frentes de tres objetivos
C://Users//Sony//workspace//ExperimentoCasoBasev2//StandardStudy//data//SPEA2//CarteraNegocios
Experiment directory exists
Experiment directory is a directory
Experiments: list of problems is shorter than the of requested threads. Creating 1
Id: 0 Partitions: 1 First: 0 Last: 0
Experiment: Number of algorithms: 3
Experiment: runs: 5
Nombre:
experimentDirectory: StandardStudy
numberOfThreads_: 1
numberOfProblems_: 1
first: 0
last: 0
Iruns: 0
latex directory: StandardStudy/latex

```

Figura 49: Ejemplo salida ejecutando la tercera opción

El cometido de esta opción es dividir la estabilidad por niveles (alta, media, baja), obtener los frentes no dominados para cada uno de estos niveles y escribir un fichero especificando los individuos correspondientes a estos frentes no dominados. Para poder dividir la estabilidad en estos tres niveles, es necesario pasarle un fichero con un frente de Pareto de tres objetivos. Para ello ha debido ejecutarse previamente, en algún momento, la opción 1 que permite la obtención de estos frentes. A continuación se evalúan los individuos pertenecientes a los frentes no dominados (cuya información se encuentra en el fichero que se acaba de obtener) con la función encargada de calcular los pares de valores rentabilidad-riesgo correspondientes a una serie mensual concreta (en nuestro caso el último dato del fichero). De esta ejecución se obtendrá una carpeta con los frentes no dominados, para cada nivel de estabilidad, y los ficheros con los valores rentabilidad-riesgo reales para la serie n.

- **La opción 4** permite modificar los parámetros del fichero de configuración mediante la entrada estándar por teclado. Esta opción mostrará un submenú con una lista de parámetros posibles a modificar. Puede hacerse uso de este submenú tantas veces

como parámetros se quieran modificar mediante esta vía. A modo de ejemplo, se muestra una traza para la modificación del número de generaciones que se utilizará como criterio de parada del algoritmo:

```
=====
                        OPTIMIZACIÓN DE CARTERAS DE INVERSIÓN
=====
Por favor, seleccione el experimento que desea ejecutar
1. Optimización de carteras robustas
2. Optimización de carteras con dos objetivos
3. Experimentos para el estudio de la estabilidad
4. Modificar parámetros del fichero de configuración
5. Salir
=====
4
=====
                        ¿Qué parámetro/s desea modificar?
=====
1. Modificar la ruta del archivo actual
2. Modificar la ruta de los frentes de pareto de referencia
3. Tamaño de la población y tamaño del archivo. Por defecto 1000 individuos
4. Número de evaluaciones. Por defecto 20000.
5. Probabilidad de mutación(entre 0.0 y 1.0
6. Probabilidad de cruce (entre 0.0 y 1.0)
7. Índice de distribución de cruce. Por defecto 20.0
8. Índice de distribución de mutación. Por defecto 20.0
9. Número de ejecuciones independientes
10. No deseo hacer más modificaciones
=====
4
Introduzca el nuevo valor
5000
=====
                        ¿Qué parámetro/s desea modificar?
=====
1. Modificar la ruta del archivo actual
2. Modificar la ruta de los frentes de pareto de referencia
3. Tamaño de la población y tamaño del archivo. Por defecto 1000 individuos
4. Número de evaluaciones. Por defecto 20000.
5. Probabilidad de mutación(entre 0.0 y 1.0
6. Probabilidad de cruce (entre 0.0 y 1.0)
7. Índice de distribución de cruce. Por defecto 20.0
8. Índice de distribución de mutación. Por defecto 20.0
9. Número de ejecuciones independientes
10. No deseo hacer más modificaciones
=====
10
No desea realizar más modificaciones

¿Desea ejecutar algún otro experimento?
=====
                        OPTIMIZACIÓN DE CARTERAS DE INVERSIÓN
=====
Por favor, seleccione el experimento que desea ejecutar
1. Optimización de carteras robustas
2. Optimización de carteras con dos objetivos
3. Experimentos para el estudio de la estabilidad
4. Modificar parámetros del fichero de configuración
5. Salir
=====
```

Figura 50: Ejemplos configuración mediante la opción 4

- **La opción 5** permite salir de la aplicación:

```
=====
                        OPTIMIZACIÓN DE CARTERAS DE INVERSIÓN
=====
Por favor, seleccione el experimento que desea ejecutar
1. Optimización de carteras robustas
2. Optimización de carteras con dos objetivos
3. Experimentos para el estudio de la estabilidad
4. Modificar parámetros del fichero de configuración
5. Salir
=====
5
Ha seleccionado la opción salir
```

Figura 51: Ejemplo salida opción 5

DATOS DE SALIDA:

Durante la ejecución del programa se muestran trazas, mediante la salida estándar (*stdout*), con información sobre el número de algoritmos, problemas y ejecuciones que se van a ejecutar, el nombre y directorio del experimento, y la ejecución y evaluación actual para el algoritmo que toque. Tras finalizar se muestra información sobre los directorios que se han creado y dónde se almacenan cada una de las métricas de desempeño calculadas.

```
=====
                        OPTIMIZACIÓN DE CARTERAS DE INVERSIÓN
=====
Por favor, seleccione el experimento que desea ejecutar
1. Optimización de carteras robustas
2. Optimización de carteras con dos objetivos
3. Experimentos para el estudio de la estabilidad
4. Modificar parámetros del fichero de configuración
5. Salir
=====
1
Experiment directory exists
Experiment directory is a directory
Experiments: list of problems is shorter than the of requested threads. Creating 1
Id: 0 Partitions: 1 First: 0 Last: 0
Experiment: Number of algorithms: 1
Experiment: runs: 5
Nombre:
experimentDirectory: ../EstudioEstabilidad/StandardStudy
numberOfThreads_: 1
numberOfProblems_: 1
first: 0
last: 0

Iruns: 0
class jmetal.metaheuristics.spea2.SPEA2
Running algorithm: SPEA2, problem: CarteraNegocios, run: 0

Poblacion creada evaluaciones: 3000
Comienza la evaluacion: 100
Comienza la evaluacion: 200
Comienza la evaluacion: 300
```

Figura 52: Salida de la ejecución de la aplicación

...


```

Ha terminado el algoritmo 0
latex directory: ../EstudioEstabilidad/StandardStudy/latex
../EstudioEstabilidad/StandardStudy/data//SPEA2/CarteraNegocios/HV
0.568669683194281
0.5955234334574565
0.633374007079595
0.6903088243464203
0.6273243888026896
../EstudioEstabilidad/StandardStudy/data//SPEA2/CarteraNegocios/SPREAD
0.6463071552643722
0.5659511479182318
0.606724809684266
0.5075622685108164
0.5663189902055344
../EstudioEstabilidad/StandardStudy/data//SPEA2/CarteraNegocios/EPSILON
0.07230062084468113
0.06148771098604598
0.05779107340458489
0.04649435046231072
0.05414028951233872
Creating ../EstudioEstabilidad/StandardStudy/latex directory
Experiment name: StandardStudy
R : ../EstudioEstabilidad/StandardStudy/R
Creating ../EstudioEstabilidad/StandardStudy/R directory
Indicator: HV
Indicator: SPREAD
Indicator: EPSILON
R : ../EstudioEstabilidad/StandardStudy/R
Indicator: HV
Indicator: SPREAD
Indicator: EPSILON

```

Figura 53: Traza mostrada durante la ejecución

Para cada experimento se crea un directorio, con el nombre que se haya especificado, con una carpeta llamada de igual modo que el experimento ejecutado. En su interior se localizan tres carpetas comunes para todos los experimentos:




 data	20/09/2010 18:18	Carpeta de archivos
 latex	20/09/2010 20:15	Carpeta de archivos
 R	20/09/2010 20:15	Carpeta de archivos

Figura 54: Estructura de la carpeta creada

- 1) Data, en cuyo interior se crea una subcarpeta por cada algoritmo ejecutado y en ella se puede encontrar una carpeta por cada problema evaluado (en nuestro caso solo 1). Dentro de esta última carpeta se localizan los ficheros con los mejores individuos y frentes de Pareto (tantos como ejecuciones se hayan realizado). Además, existe un fichero por cada métrica que se haya especificado (HV para hipervolumen, SPREAD, y EPSILON). La información de esta última métrica no se utiliza.

EPSILON	20/09/2010 20:15	Archivo	1 KB
FUN.0	20/09/2010 20:14	Archivo 0	6 KB
FUN.1	20/09/2010 20:15	Archivo 1	5 KB
FUN.2	20/09/2010 20:15	Archivo 2	6 KB
FUN.3	20/09/2010 20:15	Archivo 3	6 KB
FUN.4	20/09/2010 20:15	Archivo 4	6 KB
HV	20/09/2010 20:15	Archivo	1 KB
SPREAD	20/09/2010 20:15	Archivo	1 KB

Figura 55: Contenido de la carpeta DATA

- 2) R: con scripts para el análisis de contrastes estadísticos de las diferentes métricas de desempeño. Estos scripts no se utilizan, ya que se ha hecho uso de un script en R, adaptado a nuestros objetivos, localizado en la raíz del proyecto.
- 3) Látex: muestra tablas en látex con información estadística de las métricas de desempeño: media, mediana, desviación típica, IQR, máximo y mínimo.

Para el último experimento, se crean dos subcarpetas dentro de la carpeta *CarteraNegocios*: una con los frentes no dominados, con sus individuos y métricas, para cada nivel de estabilidad (*SolucionesNoDominadas*), y otra carpeta (*DatosReales*), que contiene los frentes de Pareto reales correspondientes a los pares de rentabilidad-riesgo para el dato n . En este subdirectorio se incluye, además de los con los valores de las métricas generadas (HV, SPREAD, EPSILON) que para este estudio carecen de significado, un archivo llamado Mahalanobis con el valor promedio entre las distancias de los frentes estimados y los previstos, para cada una de las ejecuciones realizadas.

MODO DE EJECUCIÓN CON EL SCRIPT R:

Como ya se especificó en los requisitos mínimos, para realizar contrastes estadísticos, de manera cómoda, a partir de los resultados obtenidos en las métricas de desempeño (archivos *HV*, *SPREAD* y *MAHALANOBIS*), basta con crear una carpeta que contenga el script (*contrastar.r*) y los archivos con las métricas a contrastar, con extensión *.txt*. A continuación se seleccionará el menú contextual *Archivo*, y se cambiará de directorio a la carpeta que contiene el script.

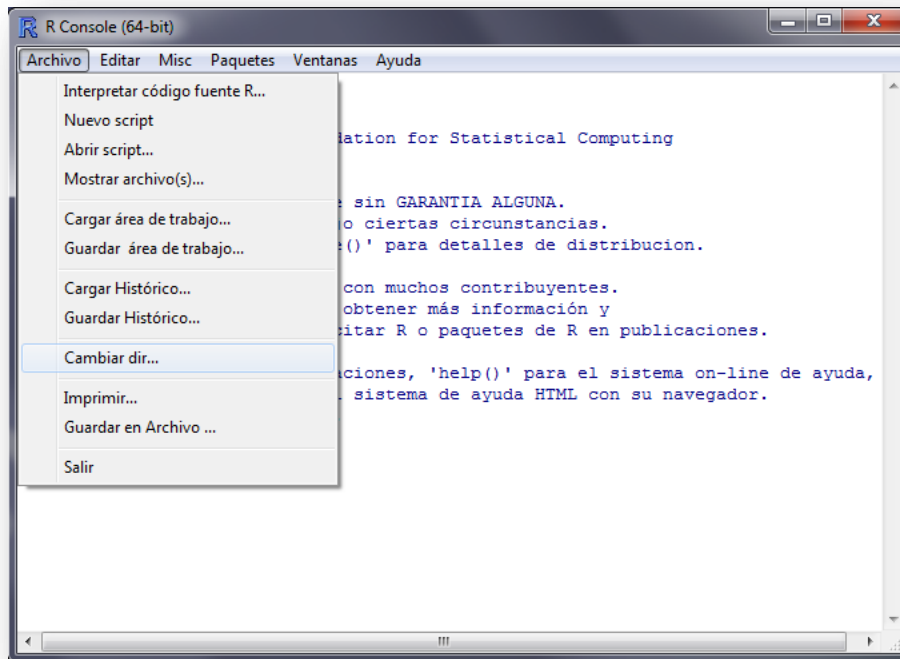


Figura 56: Menú archivo de script R

A continuación, dentro de este mismo menú contextual, se selecciona la opción Interpretar código fuente en R, escogiendo el archivo con extensión “.r”. Como resultado del análisis se obtienen dos ficheros, sin ninguna extensión: Estadísticos y Contrastes. El primero de ellos contiene información estadística de las métricas: la media, mediana, desviación típica y los valores máximo y mínimo. El segundo, contiene los resultados de los contrastes en una tabla, como se especificó en el apartado [5.1.1.7 Análisis estadístico de los resultados](#).

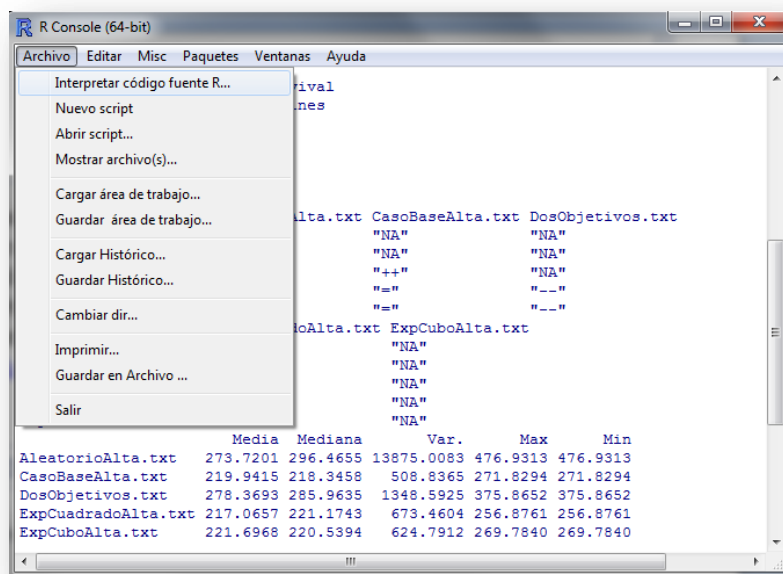


Figura 57: Menú de la aplicación de script R

EJEMPLO DE LOS FICHEROS DE SALIDA:

A continuación se muestra un ejemplo de los ficheros de salida: el primero se trata de un frente de Pareto y el otro de su individuo correspondiente:

- Fichero FUN:

5.140735813669572	0.8697764829409571
1.5943398715273565	0.7264820243209629
5.140735813669572	0.8697764829409571
5.140735813669571	0.869776482940957
4.611194556210806	0.8675298187765922
4.40390914008086	0.8652282796996269
4.126523267867575	0.8627595935675503

Figura 58: Ejemplo salida del fichero FUN

- Fichero VAR

0.2673304498919829	0.0	0.0	0.0	0.05967948980058893	0.0	0.5342711683899336
0.13871889191749448						
0.4159982451028701	0.0	0.0	0.0	0.0	0.437249450740943	0.14675230415618695
0.17110899032593035	0.0	0.0	0.0	0.05967948980058893	0.0	0.4
0.3450049013076999	0.0	0.0	0.0	0.0	0.5747066593407655	0.08028843935153462
0.20446922174223114	0.0	0.0	0.0	0.06495108605636418	0.0	0.4
0.17110899032593013	0.0	0.0	0.0	0.07003725247444116	0.0	0.3818716294955189
0.3769821277041098						
0.15250068784254744	0.0	0.0	0.0	0.09417525725365061	0.0	0.3931519995479842
0.3601720553558178						

Figura 59: Ejemplo de salida del fichero VAR

BIBLIOGRAFÍA

Referencias utilizadas para la realización del proyecto:

- [1] Guía Institucional de Carteras de Inversión, recuperado el 30/03/2010,
<http://www.slideshare.net/Seguridadsocialunesr/gua-instruccionl-de-carteras-de-inversin>
- [2] Cinco reglas para crear una buena cartera de inversión y dos ejemplos prácticos, recuperado el 11/04/2010,
<http://www.actibva.com/guias/2008/08/13-cinco-reglas-para-crear-una-buena-cartera-de-inversion-y-dos-ejemplos-practicos>
- [3] An Introduction to Investment Theory, recuperado el 23/03/2010
<http://viking.som.yale.edu/will/finman540/classnotes/notes.html>
- [4] Evaluation et Optimisation de portefeuilles d'actions
Proyecto dirigido por INRIA
- [5] Portfolio Optimization using Multi-objective Genetic Algorithms
Prisadarng Skolpadunger, Keshav Dahal and Napat Harnpornchai
- [6] H. Markowitz., Portfolio selection. Journal of Finance, vol. 7, pp. 77-91, 1952, recuperado el 11/04/2010,
<http://www.gacetafinanciera.com/TEORIARIESGO/MPS.pdf>
- [7] Teoría Moderna de Cartera, recuperado el 22/06/2010,
http://es.wikipedia.org/wiki/Teor%C3%ADa_de_Portafolio
- [8] Valoración de carteras, recuperado el 28/06/2010
http://www.stockssite.com/mc/01_Valoracion_de_Cartera.htm
- [9] Modern Portfolio Theory, recuperado el 30/06/2010
http://en.wikipedia.org/wiki/Modern_portfolio_theory
- [10] Análisis del riesgo en carteras de inversión, recuperado el 30/06/2010
<http://www.gestiopolis.com/finanzas-contaduria/estrategia-del-analisis-de-l-riesgo-de-portafolios.htm>
- [11] Iniciación a la Investigación, recuperado el 22/06/2010,
http://www.ii.uam.es/esp/posgrado/proyectos/ruben_ruiz.pdf
- [12] Optimización de portafolios accionarios a través de un micro algoritmo genético, recuperado el 21/06/2010,
http://revistas.concytec.gob.pe/scielo.php?script=sci_arttext&pid=S1810-99932007000200003&lng=es&nrm=iso
- [13] Aplicación de NSGA-II y SPEA2 para la optimización multiobjetivo de redes multicast, recuperado el 23/06/2010,
<http://redalyc.uaemex.mx/pdf/852/85201702.pdf>
- [14] Creación de Portafolios de Inversión utilizando Algoritmos Evolutivos

- Multiobjetivo, recuperado el 10/06/2010,
<http://www.cs.cinvestav.mx/Estudiantes/TesisGraduados/2005/resumenSalvadorCastroEnciso.html>
- [15] Eficiencia de Pareto, recuperado el 22/06/2010
http://es.wikipedia.org/wiki/Eficiencia_de_Pareto
- [16] Classifier Systems and Genetic Algorithms, recuperado el 21/06/2010
<http://deepblue.lib.umich.edu/bitstream/2027.42/27777/1/0000171.pdf>
- [17] Optimización de una forma hidrodinámica mediante algoritmos evolutivos, recuperado el 22/06/2010,
http://canal.etsin.upm.es/publicaciones/articulos/copinaval2005_clemente_royas_perez.pdf
- [18] A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II, recuperado el 19/06/2010,
<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=996017&userType=inst>
- [19] EMOO Web Page, recuperado el 21/06/2010,
<http://delta.cs.cinvestav.mx/~ccoello/EMOO/>
- [20] Aplicaciones de los algoritmos Evolutivos Multiobjetivo, recuperado el 22/06/2010,
<http://neo.lcc.uma.es/pdf-charlas/apli-MOEA.pdf>
- [21] Frank Schlotmann and Detlef Seese. Financial Applications of Multi-objective Evolutionary Algorithms: recent developments and future research directions. Coello-Coello, C.; Lamont, G. (eds.), World Scientific Singapore, 2004. ISBN 981-256-106-4.
- [22] Multiobjective Robustness for Portfolio Optimization in Volatile Environments Ghada Hassan y Christopher Clack
- [23] Gestión de Proyectos, recuperado el 24/06/2010,
<http://www.knoow.net/es/cieeconcom/gestion/gestiondeproyectos.htm>
- [24] Web Oficial de la herramienta ParadisEO, recuperado el 23/03/2010,
<http://paradiseo.gforge.inria.fr/>
- [25] Web Oficial del framework de JMetal, recuperado el 30/04/2010,
<http://jmetal.sourceforge.net/>
- [26] A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II
- [27] Mochon Morcillo, Francisco y APARICIO, Rafael I., Diccionario de términos financieros y de inversión, 2ª Ed., (Madrid, Mc Graw-Hill, 1998), 502 págs., pág. 330.

GLOSARIO

La intención de este glosario es ofrecer un conjunto de definiciones breves de los términos financieros utilizados en el desarrollo de este documento.

Acciones. Representan una parte del capital de una empresa por lo que son una participación de su propiedad; otorga a sus tenedores (socios) el derecho a participar de los beneficios de la empresa (por ejemplo: dividendos). No tienen fecha de vencimiento. Requieren de un largo plazo para adquirir mayor madurez del rendimiento. Estos títulos son presentados a los inversores a través de la Bolsa de Valores.

Activos financieros. Conjunto de bienes y derechos pertenecientes a una persona jurídica o física. Dentro del balance de una empresa identifica los bienes y derechos de que es titular.

Aversión al riesgo. Es la posición de un inversor que no desea someter sus inversiones financieras a altos riesgos, por lo que sus inversiones serán muy selectivas considerando siempre la eliminación del riesgo aunque alcance una menor rentabilidad.

Banco. Es una institución que capta recursos del público y presta dichos recursos a plazos que no tienen porqué coincidir con los términos a los cuales los captó; transforma los activos financieros.

Bursatilidad. Indicador de los niveles de compra o venta de una acción, a través del volumen de acciones que se intercambia diariamente.

Cartera. La cartera de inversión la componen los distintos activos financieros y bienes tangibles en los que invertimos nuestro dinero en busca de una rentabilidad.

Composición. Distribución de activos dentro de una cartera de inversión.

Índice de precios y cotizaciones. Este índice es un indicador del comportamiento del mercado accionario (renta variable). Es, en términos breves, un promedio ponderado de los precios de las acciones de un conjunto de emisoras que se consideran las más representativas del mercado que se esté tratando.

Entidades Emisoras: Entidad pública (organismo) o privada (empresa) que coloca algún activo o pasivo referido como valor en el mercado de valores (emisión), para ello debe estar representada por una casa de bolsa.

Inversión. Se realiza cuando sacrificamos el consumo presente a cambio de una promesa de un consumo futuro de mayores beneficios.

Inversor: personas físicas o morales, nacionales o extranjeras que a través de una casa de bolsa colocan sus recursos a cambio de valores, para obtener rendimientos.

Autoridades y Organismos Autorregulatorios: Fomentan y supervisan la operación ordenada del mercado de valores y sus participantes conforme a la normatividad vigente.

Valores. Acciones, títulos, bonos, papel comercial, deuda, obligaciones, etc.

ELEMENTOS BÁSICOS DE UNA INVERSIÓN:

Rendimiento: es la ganancia que se obtiene al realizar una inversión y, normalmente, se obtiene después de cumplirse uno o varios plazos de tiempo. El rendimiento se ve disminuido al aplicarle impuestos y comisiones. Se dice que a mayor plazo se obtienen mayor madurez de la inversión, lo que genera mayor rendimiento. Se puede otorgar de dos formas:

- 1) Dividendos (intereses) que genera la emisora.
- 2) Ganancias o pérdidas de capital, que es el diferencial entre el precio de compra y el precio de venta del valor y/o amortizaciones.

Riesgo: es la posibilidad que siempre existe de que se pueda obtener poco o ningún rendimiento, e incluso disminuir o perder la inversión original. Eventos económicos, políticos, sociales y naturales pueden influenciar el comportamiento de las inversiones, generando desempleo, desequilibrios fiscales, económicos y de la cuenta externa, etc., complicando el funcionamiento de los mercados financieros, cambiando con ello las expectativas de rentabilidad y riesgos de las carteras, y de los flujos efectivos.

Plazo: es el período de tiempo durante el cual se realiza una inversión, desde su compra hasta su venta o vencimiento; lapso en el que normalmente no se puede disponer de lo invertido.

Liquidez: es la facilidad con la que se puede convertir la inversión en dinero, particularmente durante períodos en los que se están observando incrementos sustanciales en las tasas de interés. Esto es por ejemplo, vendiendo los instrumentos financieros a un tercero (en el mercado de valores) aunque se accede a un menor rendimiento, debido a que se comparte éste con el tercero.

Diversificación: consiste en repartir y compensar el riesgo, mediante la adquisición de instrumentos financieros con diferentes rendimientos, riesgo, plazos y liquidez, con el fin de armar una cartera de instrumentos financieros que en promedio prometa un rendimiento y riesgo acordes con los requeridos por el inversor. Se diversifican las inversiones operando instrumentos financieros en los distintos mercados, con diferentes niveles de liquidez, entre diferentes emisoras de sectores distintos, en divisas, en bienes raíces, en metales, en diferentes inversiones financieras, etc. Para equilibrar las pérdidas y ganancias con el fin de diluir las bajas coyunturales del mercado o de las propias acciones.

División del riesgo: resulta conveniente dividir el riesgo total de un instrumento financiero (la desviación estándar de sus rendimientos) en una parte que pertenece a esa empresa en particular que puede ser eliminado por diversificación, y otra que pertenece al mercado que no es diversificable. El primero se conoce como riesgo no sistemático del instrumento financiero y el segundo como su riesgo sistemático o de mercado, el cual caracteriza al sistema general o entorno de la empresa. Estos mismos términos se aplican para referirse al riesgo total de una cartera.

Riesgos y rentabilidad en el mercado: se dice que los mercados son eficientes cuando los precios de los valores se ajustan con mucha rapidez debido a que reflejan plenamente toda la información disponible.

Instrumentos financieros: la inversión de los recursos se puede realizar en diferentes tipos de instrumentos financieros, como bancarios, gubernamentales o bursátiles, así como en divisas.

Los instrumentos bancarios especifican la tasa de rendimiento que pagarán al vencimiento de un plazo específico. Son inversiones seguras, pero toda inversión posee un nivel de riesgo, esto sin contar que existen algunos intermediarios financieros que no otorgan ninguna garantía al inversor en caso de quiebra.

Cualquier inversión no está libre de riesgo. Siempre existe la posibilidad del no pago por parte del emisor, debido normalmente a los constantes, y en algunos casos abruptos, cambios en la economía que afectan a las tasas de interés y en consecuencia el rendimiento esperado. El único emisor con riesgo cero de no pago (en moneda nacional) es el Gobierno.

Los instrumentos bursátiles son títulos o bonos (valores) que se compran y venden dentro del mercado de valores. Así se clasifican en tres categorías:

- 1) Instrumentos de deuda a corto plazo
- 2) Instrumentos de deuda a largo plazo
- 3) Las acciones