CAPÍTULO 3

METODOLOGÍA

El contenido de este capítulo abarca la metodología que se siguió en el desarrollo del algoritmo. En la sección 3.1 se da la descripción de la planeación de la aplicación, de la determinación de la interfaz y de la extracción de la base de datos.

1.1 Descripción de la Metodología

Como en todo proceso de programación, se seguirán las fases necesarias para encontrar la solución del problema. Estas fases incluyen lo siguiente:

- Determinar las necesidades para desarrollar el algoritmo.
- Planear la aplicación que satisfaga estas necesidades.
- Determinar la interfaz más apropiada para el problema.
- Extraer la base de datos necesaria.
- Crear fórmulas, codificar las macros, y construir la interfaz del usuario.
- Documentar el desarrollo de la aplicación.
- Compilar y ejecutar el programa.
- Verificar y depurar.
- Poner al día la aplicación en caso necesario.

3.1.1 Desarrollo del Algoritmo

El primer paso del proceso es establecer lo que se requiere para desarrollar el algoritmo. Antes que nada, se debe tomar en cuenta que el problema que se trata de resolver es un problema de minimización. En este tipo de problemas, el obstáculo básico para encontrar el mínimo global es la existencia de mínimos locales. De acuerdo al artículo de Maringer y Kellerer, el máximo problema que se enfrenta al seleccionar los activos de una cartera, es el de que los métodos iterativos usuales de búsqueda pueden quedar atrapados en un vecindario con un mínimo local, debido a que el cambio a otro vecindario puede significar un aumento temporal en el valor de la función objetivo que se está tratando de minimizar (Maringer & Kellerer, 2003). Por este motivo, el algoritmo que aquí se presenta combina las características del algoritmo de Templado Simulado y las estrategias evolutivas.

Con respecto al planteamiento del modelo, a continuación se definirán las restricciones que se utilizan en esta tesis. Como se mencionó en la sección 2.5, Perold enriqueció el modelo de Markowitz introduciendo ciertas restricciones que reflejan mejor la complejidad del mundo financiero real (Perold, 1984). Las siguientes son las restricciones totales propuestas en su modelo:

- **Restricciones de presupuesto y de rendimiento esperado:**

  Estas restricciones ya se encuentran en el modelo básico de Markowitz.

- **Restricciones de límite inferior y superior:**

  Definen los límites de las proporciones de cada activo que serán incluidas en el portafolio. Estas restricciones pueden reflejar restricciones institucionales para la composición de la cartera.

- **Restricciones superiores de comercio:**

  Define los límites superiores en la variación de las cantidades de cada activo contenido en la cartera de un período al siguiente. Aquí, $x_i^{(0)}$ denota el peso del activo *i* en el portafolio inicial, $\overline{B_i}$ denota la compra máxima y $\overline{S_i}$ denota la venta máxima del activo *i* durante el período actual ($\forall i \in \{1,...,n\}$). Es decir, tanto la venta como la compra que se haga de cada activo en este período deben ser menor al límite de compra $\overline{B_i}$ o menor al límite de venta $\overline{S_i}$ respectivamente. Estas restricciones pueden modelar costos por transacción, como pueden ser los impuestos, comisiones, efectos de falta de liquidez, etc.

- **Restricciones inferiores de comercio:**

  Define límites inferiores en la variación de las cantidades de cada activo contenido en la cartera de un período al siguiente. Dichos límites pueden reflejar el hecho de que un inversionista no pueda, o no quiera, modificar su cartera comprando o vendiendo pequeñas cantidades de activos, debido a que los contratos de transacción requieren de volúmenes significantes o debido a la existencia de costos fijos relativamente altos relacionados a cada transacción. Así para el activo *i*, se tienen tres opciones para el siguiente período, o las proporciones no cambian, o una mínima cantidad $\underline{B_i}$ debe ser comprada, o una mínima cantidad $\underline{S_i}$ debe ser vendida.

- **Restricción de número máximo de activos:**

  Define el número máximo *N* de activos que pueden ser incluidos en la cartera.

Aumentando estas restricciones al modelo básico, se obtiene la siguiente formulación matemática:

Min.	$\sum_{i=1}^{n}\sum_{j=1}^{n}\sigma_{ij}x_i x_j$	Función objetivo[1]	(3.1)
s.t.	$\sum_{i=1}^{n} r_i x_i = R_{esp}$	Restricción de Rendimiento Esperado	(3.2)
	$\sum_{i=1}^{n} x_i = 1$	Restricción de Presupuesto	(3.3)
	$\underline{x_i} \le x_i \le \overline{x_i}$	Restricciones de Límite Inferior y Superior	(3.4)
	$\max\left(x_i - x_i^{(0)}, 0\right) \le \overline{B_i}$	Restricción Superior de Compra	(3.5)
	$\max\left(x_i^{(0)} - x_i, 0\right) \le \overline{S_i}$	Restricción Superior de Venta	(3.6)
	$x_i = x_i^{(0)}$ ó $x_i \ge \left(x_i^{(0)} + \underline{B_i}\right)$	Restricción de Comercio	
	ó $x_i \le \left(x_i^{(0)} - \underline{S_i}\right)$		(3.7)
	$\left\lvert\left\{i \in \{1,\ldots,n\} : x_i \ne 0\right\}\right\rvert \le N$	Restricción de Número Máximo de Activos	(3.8)
	$\forall i \in \{1,\ldots,n\}$		

Para resolver este modelo, se usará como base el algoritmo de Maringer et al. (ver sección 2.4 de esta tesis). La idea general de su algoritmo es empezar con la técnica del Templado Simulado, escoger arbitrariamente una solución inicial y modificarla permitiendo cambios aleatorios cuando los cambios resulten en un mejoramiento de la

[1] Recordar que x_i representa la proporción del activo *i* en el portafolio y que C es la matriz *n* x *n* de las covarianzas de los *n* rendimientos

calidad de la solución. Adicionalmente, para no quedar atrapado en óptimos locales, cambios que no producen mejoramiento, son aceptados de acuerdo al criterio de Metrópolis (ver sección 2.3 de esta tesis) con una probabilidad decreciente en cada iteración. Sin embargo, a diferencia del algoritmo estándar de Templado Simulado donde una sola solución es considerada, en esta versión se usa toda una población de soluciones, de las que las peores son eliminadas y reemplazadas, ya sea por un clon de una de las mejores soluciones, o por un nuevo individuo producto de las mejores soluciones anteriores.

Específicamente, el algoritmo empieza con la inicialización aleatoria de las soluciones que representan a cada portafolio. Esto incluye seleccionar *N* de los *n* activos disponibles y asignarles pesos aleatorios ($x_i^{(0)} \forall i \in \{1,...,n\}$). Las siguientes iteraciones consisten de tres pasos.

Paso 1: Modificación de la estructura de la cartera.

Paso 2: Evaluación y ordenamiento de las soluciones según su calidad.

Paso 3: Reemplazo de las soluciones más pobres de la población.

Modificación: Para cada solución *c*, d_N activos de la solución de la cartera P_c son seleccionados. Para estos activos, los respectivos cambios en los pesos son hechos de acuerdo a $x_i^{'} = \max\{x_i + \tilde{z}_i;0\}$ donde $\tilde{z}_i \in [-U_t;U_t]$, los pesos de los activos restantes permanecen sin cambios, es decir $x_i^{'} = x_i$. Por su parte, U_t indica la amplitud de movimiento de los cambios de la iteración *t*. Si $x_i^{'}$ se vuelve cero, con una probabilidad p_r el respectivo activo es reemplazado por un nuevo activo *j* que no esté aún incluido en la

cartera $(x_i = 0)$. Al activo *j* se le asigna un peso aleatorio $x_i^{'} \in [0;2U_t]$. Con probabilidad $1 - p_r$, el activo *i* permanece en la cartera con el peso 0, es decir, el portafolio contendrá entonces menos de *N* activos.

Una vez cambiados los pesos, se estandarizan de manera que la restricción $\sum_{i=1}^{n} x_i = 1$ se cumpla. Después se calcula la calidad del portafolio modificado $P^{'}$. De acuerdo a los principios del algoritmo de Templado Simulado (ver sección 2.3), las modificaciones son aceptadas con una probabilidad $p = \min\left\{0; \exp\left(\frac{E_i - E_j}{c(t)}\right)\right\}$. Finalmente, para cada solución se sigue el mismo procedimiento de generar una cartera modificada y decidir si se acepta el cambio o no, un número fijo de veces.

Evaluación: Las soluciones son ordenadas de acuerdo a su calidad. Esta evaluación y ordenamiento es crucial para tomar la decisión acerca de cuales soluciones retener y cuales desechar. Siguiendo el modelo de Maringer (Maringer & Kellerer, 2003), sólo π soluciones serán seleccionadas para ser padres en la siguiente generación. A estas soluciones se les da el nombre de *pródigos*. De acuerdo al lugar que ocupen en el ordenamiento, a las carteras prodigio se les asigna un factor de ordenamiento linealmente decreciente que va de $\pi + 1$ a 1. Además, al grupo de soluciones generadoras se le suman ε soluciones elitistas, es decir, las mejores soluciones encontradas hasta ese momento. Esto es hecho asignando a la mejor cartera encontrada hasta ahora un factor de ordenamiento ε y sumándola a la lista de pródigos. Las soluciones de la población actual que no son pródigos, tienen un factor de ordenamiento igual a 0.

Reemplazo: Para lograr por una parte retener las mejores características de los pródigos y por otra parte, eliminar las peores, las *w* soluciones más pobres en cuanto a su calidad son reemplazadas por soluciones consideradas de mayor potencial. El reemplazo se hace siguiendo el procedimiento siguiente: se aplica el método de selección de la ruleta (ver sección 2.2) y se escoge un método de reemplazo para el portafolio ineficiente. Se crea una réplica (clon) sin cambios de la cartera seleccionada con una probabilidad p_c, y con probabilidad $1-p_c$ se utiliza otro método de reemplazo llamado "Averaged Idol". Este método consiste en calcular un peso promedio para cada activo *i*, basado en las carteras de la lista elitista y de pródigos. Los pesos x_i de cada activo *i* en la solución *c*, se multiplican por sus factores de ordenamiento, se suman y se normalizan de manera que la suma de todos sea 1. Los pesos promediados son usados para las probabilidades de seleccionar *N* activos y asignarles pesos que reflejen estos pesos promediados. El sacar los pesos promediados provoca que un activo sea preferido cuando se encuentra en varias carteras de la lista de elitistas con un peso alto.

El algoritmo termina después de un número fijo de iteraciones y se reporta la mejor solución encontrada, es decir, el último portafolio elitista.

Para reportar los resultados, se tomarán varios valores para R_{esp}, se correrá el algoritmo y se presentará el valor del vector *X* de los pesos de los activos y el riesgo total del portafolio para cada uno. Con estos valores se mostrará la gráfica de la frontera de media-varianza. Junto con los valores obtenidos se presentará también el tiempo requerido por el algoritmo para llegar al resultado.

3.1.2 Planeación de la Aplicación

Teniendo en cuenta las necesidades del problema, para el desarrollo del algoritmo se requiere de una aplicación que cumpla con las siguientes características:

- Debe proporcionar la solución apropiada al problema.
- Usa métodos apropiados y eficaces y algoritmos para lograr su trabajo.
- Maneja una interfaz del usuario clara y consistente.
- Permite la buena documentación de fórmulas, macros y elementos de la interfaz, de manera que se permitan los cambios subsecuentes, si son necesarios.
- Está diseñada para que pueda modificarse de manera simple, sin tener que hacer cambios mayores.

3.1.3 Determinación de la Interfaz

Una aplicación que reúne estas características es Visual Basic for Applications. Visual Basic for Applications (VBA) es un lenguaje de programación lanzado por Microsoft en 1991. VBA fue construido a partir de una aplicación anterior llamada BASIC (siglas de Beginner's All- purpose Symbolic Instruction Code, Código de Instrucciones Simbólicas de todo Propósito para Principiantes. Basic tuvo éxito rápidamente y empezó a mejorar y evolucionar con el paso de los años. Excel 5 fue la primera aplicación en el mercado en ofrecer Visual Basic for Applications y hoy es conocida como la aplicación de lenguaje scripting más común de Microsoft (Walkenbach, 2002).

Usando VBA, se pueden crear funciones personalizadas en la hoja de cálculo para simplificar fórmulas y cálculos. Además, se pueden acceder datos importantes directamente desde el ambiente de la hoja de cálculo. El éxito de VBA radica en entender el modelo de objetos para cada aplicación que se requiere. Por esta razón, se decidió utilizar VBA para desarrollar el programa computacional que obtendrá los resultados del modelo.

3.1.4 Base de Datos

La base de datos necesaria para probar el algoritmo se obtendrá a partir de los precios semanales de los activos seleccionados. Después se calcularán los rendimientos semanales de cada activo *i*. Los rendimientos semanales se obtienen con la fórmula siguiente:

$$\mathrm{Re}\,n\,\mathrm{dim}\,iento = \frac{\mathrm{Pr}\,ecioActual - \mathrm{Pr}\,ecioAnterior}{\mathrm{Pr}\,ecioAnterior}$$

Posteriormente se calcula la matriz de covarianzas de los rendimientos. Teniendo el rendimiento promedio y la matriz de covarianzas se puede calcular el rendimiento esperado y la varianza del total del portafolio.