Robust Optimization of Algorithmic Trading Systems

TESIS DOCTORAL

José Manuel Berutich Lindquist

Departamento de Lenguajes y Ciencias de la Computación
Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

Mayo 2017

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA

AUTOR: José Manuel Berutich Lindquist

http://orcid.org/0000-0002-0918-9634

EDITA: Publicaciones y Divulgación Científica. Universidad de Málaga

BY NC ND

Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-SinObraDerivada 4.0 Internacional:
http://creativecommons.org/licenses/by-nc-nd/4.0/legalcode
Cualquier parte de esta obra se puede reproducir sin autorización
pero con el reconocimiento y atribución de los autores.
No se puede hacer uso comercial de la obra y no se puede alterar, transformar o hacer obras derivadas.

Esta Tesis Doctoral está depositada en el Repositorio Institucional de la Universidad de Málaga (RIUMA): riuma.uma.es

UNIVERSIDAD
DE MÁLAGA

Documento maquetado con TEXIS v.1.0+.

UNIVERSIDAD DE MÁLAGA

Robust Optimization of Algorithmic Trading Systems

Memoria que presenta para optar al título de Doctor en Informática

José Manuel Berutich Lindquist

Programa de Doctorado

Ingeniería del Software e Inteligencia Artificial

Dirigida por los Doctores

Francisco Luna Valero
Francico López Valverde

Departamento de Lenguajes y Ciencias de la Computación
Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

Mayo 2017

UNIVERSIDAD DE MÁLAGA

Copyright © José Manuel Berutich Lindquist

UNIVERSIDAD DE MÁLAGA

Departamento de Lenguajes y Ciencias de la Computación
Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

El Doctor Francisco López Valverde, profesor titular y el Doctor Francisco Luna Valero, profesor contratado, ambos pertenecientes al Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga, certifican:

que D. José Manuel Berutich Lindquist, Ingeniero en Informática por Bentley University (Boston, EE.UU.), ha realizado en el Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga, bajo dirección de ambos, el trabajo de investigación correspondiente a su Tesis Doctoral titulada:

Robust Optimization of Algorithmic Trading Systems

Revisado el presente trabajo, estimamos que puede ser presentado al tribunal que ha de juzgarlo. Y para que conste a efectos de lo establecido en la legislación vigente, autorizamos la presentación de la Tesis Doctoral en la Universidad de Málaga.

En Málaga, 22 de Mayo de 2017

Dr. Francisco López Valverde Dr. Francisco Luna Valero

[illegible] [illegible]

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD DE MÁLAGA

Abstract

GAs *(Genetic Algorithms*) and GP (*Genetic Programming*) are investigated for finding robust *Technical Trading Strategies* (TTSs). TTSs evolved with standard GA/GP techniques tend to suffer from over-fitting as the solutions evolved are very fragile to small disturbances in the data. The main objective of this thesis is to explore optimization techniques for GA/GP which produce robust TTSs that have a similar performance during both optimization and evaluation, and are also able to operate in all market conditions and withstand severe market shocks.

In this thesis, two novel techniques that increase the robustness of TTSs and reduce over-fitting are described and compared to standard GA/GP optimization techniques and the traditional investment strategy *Buy & Hold*. The first technique employed is a robust multi-market optimization methodology using a GA. Robustness is incorporated via the environmental variables of the problem, i.e. variablity in the dataset is introduced by conducting the search for the optimum parameters over several market indices, in the hope of exposing the GA to differing market conditions. This technique shows an increase in the robustness of the solutions produced, with results also showing an improvement in terms of performance when compared to those offered by conducting the optimization over a single market.

The second technique is a random sampling method we use to discover robust TTSs using GP. Variability is introduced in the dataset by randomly sampling segments and evaluating each individual on different random samples. This technique has shown promising results, substantially beating *Buy & Hold*.

Overall, this thesis concludes that *Evolutionary Computation* techniques such as GA and GP combined with robust optimization methods are very suitable for developing trading systems, and that the systems developed using these techniques can be used to provide significant economic profits in all market conditions.

UNIVERSIDAD
DE MÁLAGA

UNIVERSIDAD DE MÁLAGA

Resumen

Cuando las leyes de la matemática se refieren a la realidad, no son ciertas; cuando son ciertas, no se refieren a la realidad.

Albert Einstein

I Introducción

En las dos últimas décadas, el AT (*Algorithmic Trading*) utilizando EC (*Evolutionary Computation*) ha atraído mucha atención de investigadores académicos tanto del ámbito de las finanzas, como del soft-computing (Aguilar-Rivera et al., 2015; Hu et al., 2015). AT es un término comúnmente usado para describir programas informáticos que automatizan una o más etapas del proceso de negociación de activos en los mercados financieros.

En la actualidad, los sistemas de AT manejan aproximadamente del 50 % al 60 % de todas las acciones negociadas en los Estados Unidos y la Unión Europea y es una fuente importante de innovación en computación, especialmente en aprendizaje computacional y en computación distribuida (Nuti et al., 2011; Hendershott and Riordan, 2013). Estos sistemas se despliegan en mercados muy líquidos en clases de activos tales como acciones, futuros, derivados, bonos y divisas.

Los principales objetivos que esta tesis trata de abordar, son el diseñar y optimizar TTSs (*Technical Trading Strategies*) robustas, dos problemas clásicos en el ámbito del AT. Una TTS consiste en una serie de reglas o condiciones que determinan si se compra o se vende un determinado instrumento financiero. Para abordar ambos problemas utilizamos técnicas de EC a las que incorporamos métodos de optimización robusta.

La optimización y diseño de una TTS es muy similar a un problema de aprendizaje computacional, donde estamos tratando de aprender el mejor conjunto de parámetros de un modelo sobre un conjunto de datos histórico. Para determinar cómo funcionarían las reglas obtenidas durante la optimización en el mundo real, necesitamos dividir los datos en dos subconjuntos uno para entrenamiento y otro evaluación.

UNIVERSIDAD DE MÁLAGA

Uno de los mayores problemas al aplicar técnicas de EC es que hacen tan buen trabajo en resolver problemas multimodales que son atraídas a zonas de fitness alta pero que no son indicativas de soluciones robustas. Esto genera soluciones que obtienen un rendimiento muy alto durante el proceso de optimización, pero que cuando son evaluadas en datos nuevos, obtienen resultados muy pobres.

En este trabajo de tesis doctoral nos planteamos las siguientes hipótesis:

- ¿Podemos usar técnicas optimización robusta para generar estrategias de trading que obtengan un rendimiento aceptable cuando sean comprobadas con nuevos datos?
- ¿Incrementa las robustez de las soluciones el incorporar al proceso de optimización otras series temporales de diferentes instrumentos financieros?

I.I Contribuciones

La siguiente lista resume las principales aportaciones que se han producido durante esta tesis doctoral:

- La robustez es incorporada al optimizar los parámetros de una estrategia de trading sobre múltiples mercados. Usamos la media y la desviación típica de la función objetiva original como nuevos objetivos robustos de un problema de optimización multiobjetivo.
- Emplear múltiples mercados expone al algoritmo de optimización a ruido y variabilidad, lo que conlleva a soluciones robustas que obtienen un rendimiento similar al obtenido durante el proceso de optimización cuando son comprobadas sobre datos nuevos.
- Se demuestra la utilizad de incorporar la "Perdida Máxima" como un objetivo adicional a minimizar.
- Un mecanismo de muestreo aleatorio para dividir las series temporales que no requiere la intervención del usuario ni de ningún método de aprendizaje computacional no supervisado.
- El uso de una función de fitness robusta que calcula el fitness medio sobre muestras aleatorias del conjunto de entrenamiento y que reduce el sobre-ajuste de las soluciones.
- El uso de diferentes métricas derivadas del análisis técnico y cuantitativo como la media de los retornos, el alpha y beta derivado del CAPM (*Capital Asset Pricing Model*), el ratio de Sharpe y la volatilidad calculadas con diferentes duraciones, que son empleadas para generar TTSs robustas.

UNIVERSIDAD DE MÁLAGA

I.II Organización

Esta tesis se organiza de la siguiente manera:

- **Capítulo 1 - Introducción**
  Este capítulo presenta el tema de esta investigación, contribuciones, alcance y publicaciones que apoyan esta tesis.

- **Capítulo 2 - Metaheurísticas**
  Este capitulo introduce las técnicas metaheurísticas para la optimización mono y multiobjetivo y exploramos más detalle las técnicas de computación evolutiva que serán utilizadas a lo largo de esta tesis.

- **Capítulo 3 - Optimización Evolutiva Robusta**
  Este capítulo presenta la optimización robusta en el contexto de los algoritmos evolutivos y cómo los conceptos más significativos extraídos de la literatura pueden ser incorporados a los problemas de optimización de uno o varios objetivos.

- **Capítulo 4 - Trading Algorítmico**
  Este capítulo hace una introducción general del trading algorítmico y sus diferentes componentes. Se presentan las metodologías de análisis financiero utilizadas como base para las estrategias de trading desarrolladas en esta tesis. Se analiza en más detalle el análisis técnico y se muestran los indicadores técnicos junto a algunos otras métricas derivadas del análisis cuantitativo empleadas.

- **Capítulo 5 - Optimización Robusta de Estrategias de Trading Técnicas**
  Este capítulo presenta una metodología de optimización robusta de TTSs sobre múltiples mercados empleando un GA (*Genetic Algorithm*) donde la robustez se incorpora a través de las variables ambientales del problema. La búsqueda de los parámetros óptimos se lleva a cabo en varios mercados. Los trabajos más significativos son analizados, a continuación se define formalmente el problema de optimización resuelto, y se continúa explicando el enfoque algorítmico y la metodología empleada en nuestra experimentación junto con los resultados obtenidos más relevantes.

- **Capítulo 6 - Diseño y Optimización Robusto de Estrategias de Trading Técnicas**
  Este capítulo extiende el trabajo del capítulo anterior, pero esta vez utilizando GP (*Genetic Programming*) para diseñar y optimizar TTSs robustas que se utilizan para gestionar una cartera de acciones de la bolsa española. En primer lugar, introducimos el tema y continuamos discutiendo los trabajos de investigación más relevantes en el área. A

UNIVERSIDAD DE MÁLAGA

continuación, el problema se describe junto con el enfoque algorítmico y la metodología empleada para la experimentación realizada con objeto de comprobar nuestras hipótesis de investigación.

- **Capítulo 7 - Conclusiones y trabajo futuro**
  Este capítulo presenta las conclusiones más importantes que se han extraído durante el trascurso de este trabajo de investigación y las líneas propuestas de trabajo futuro.

II Conceptos Básicos

En la primera parte de la tesis hacemos una revisión a los conceptos fundamentales y el estado del arte de la metaheurística, la optimización robusta evolutiva y el trading algorítmico

II.I Metaheurísticas

Las metaheurísticas son métodos de búsqueda y optimización que combinan procedimientos de mejora local con estrategias de alto nivel normalmente estocásticas para crear un proceso capaz de escapar de óptimos locales y realizar una búsqueda eficiente en espacios de búsqueda que pueden ser muy extensos. (Glover, 1986). Las metaheurísticas pueden aplicarse a una gran variedad de problemas de optimización, ya que son métodos genéricos que incorporan información específica sobre el problema en cuestión (representación de las soluciones, operadores, etc.).

Tradicionalmente las metaheurísticas pueden clasificarse en técnicas basadas en población o basadas en trayectoria. Las técnicas basadas en población mejoran un conjunto de soluciones mientras las basadas en trayectoria una única solución. Existen otras clasificaciones como las técnicas basadas o no en la naturaleza, o técnicas que hacen uso o no de memoria.

Esta trabajo de tesis utiliza EC. La EC es una familia de algoritmos metaheurísticos muy capacitada para resolver problemas en el ámbito de la ingeniera financiera y el AT (Iba and Aranha, 2012). La EC se basa en los principios neo-darwinianos de la evolución, donde una población de múltiples soluciones candidatas se evolucionan utilizando procesos biológicamente inspirados como la reproducción, mutación, recombinación y selección. La EC no hace ninguna suposición acerca del paisaje de fitness subyacente, esta generalidad le permite aproximar soluciones a todo tipo de problemas. Las técnicas de EC que tratamos en esta tesis se describen a continuación.

Genetic Algorithms

Los GAs son técnicas de búsqueda y optimización inspiradas en la evolución natural y la genética. Los GAs son técnicas muy eficaces para resolver

UNIVERSIDAD DE MÁLAGA

problemas donde el espacio de búsqueda es muy extenso, complejo y multimodal. Esta es la motivación de usar este tipo de algoritmo para resolver los problemas de optimización de uno y de varios objetivos en el capítulo 5.

El funcionamiento general de un algoritmo genético empieza por la representación de las soluciones del problema. Los cromosomas codifican estas soluciones en una estructura de datos similar a una cadena, parecida al código ADN de los organismos vivos. Un GA típicamente comienza con una población de individuos o cromosomas generada al azar. A continuación, cada individuo es evaluado, y se obtiene una medida de fitness que determina la calidad de las soluciones codificadas como cromosomas. Los individuos son seleccionados probabilísticamente para su reproducción. Durante la selección, los individuos con mejor fitness tienen una mayor probabilidad de reproducirse que otros miembros de la población que pueden tener un valor de fitness peor. Después se aplica el cruce y la mutación a los individuos seleccionados previamente. El operador de cruce mezcla los cromosomas de los padres para formar nuevos hijos, mientras que la mutación altera parte del cromosoma del individuo. Este proceso continúa durante varias generaciones o hasta que se cumple algún criterio de parada.

Genetic Programming

La GP (Koza, 1992) es una técnica de EC para resolver problemas automáticamente sin requerir que el usuario especifique o sepa la forma de la solución de antemano. De un modo abstracto, GP es un método para conseguir que las computadoras resuelvan problemas automáticamente empezando por una orden de alto nivel de lo que se debe hacer. La GP es similar a un GA, pero tiene una diferencia significativa en la forma en que se representan las soluciones. Mientras que un GA emplea una codificación de cadena de longitud fija, GP emplea una representación de longitud variable en forma de árboles de sintaxis. Las variables y constantes del programa se denominan los terminales y son las hojas del árbol Los funciones son los nodos internos y normalmente pueden ser funciones aritméticas o lógicas, o cualquier otro tipo necesaria para resolver el problema en cuestión.

Para resolver los problemas abordados en este trabajo de tesis doctoral, empleamos los siguientes tipos de algoritmos evolutivos:

- **Mono-objetivo**:
  - Algoritmo Genético
  - Programación Genética

- **Multi-objetivo**:
  - Algoritmo Genético basado en NSGA-II

UNIVERSIDAD DE MÁLAGA

II.II Optimización Evolutiva Robusta

El término “robusto” tiene muchas definiciones dependiendo del autor, pero puede definirse de manera general como la capacidad de un sistema para preservar su funcionalidad a pesar de las perturbaciones. Las soluciones de problemas de optimización del mundo real deben ser robustas ya que su rendimiento no debe verse afectado por pequeños cambios en las variables de diseño o las condiciones ambientales. Esto no suele considerarse en los algoritmos de optimización tradicionales que asumen estas variables como constantes. La robustez y el rendimiento pueden ser objetivos contrapuestos, por lo que el objetivo de las técnicas de optimización robusta no sólo debe maximizar el rendimiento sino también garantizar una robustez suficiente.

La robustez se suele medir después de la optimización realizando un análisis de sensibilidad, que mide la respuesta de las soluciones optimizadas a pequeñas variaciones de parámetros. Dado que las soluciones optimizadas considerando sólo el rendimiento y no teniendo en cuenta su variación (robustez) podrían no incluir los mejores resultados, se debe realizar un análisis de robustez durante la búsqueda (Beyer and Sendhoff, 2007; Gaspar-Cunha and Covas, 2008), lo que puede lograrse introduciendo una medida de robustez durante la optimización que reemplace la función objetivo original con una expresión que mida tanto el rendimiento como la robustez o afrontando el problema con un enfoque multiobjetivo donde se busca optimizar el rendimiento y robustez.

Los principales conceptos que se pueden extraer de la literatura son que se puede incorporar robustez abordando el problema como un problema monoobjetivo, donde se utilizaría el fitness medio calculado en el vecindario de los parámetros. Alternativamente, el problema se puede abordar con un enfoque multiobjetivo done la función objetiva original se trasforma en dos, el fitness medio y la desviación típica (u otra medida de dispersión estadística como la varianza) medidas en el vecindario de los parámetros

Uno de los objetivos de este trabajo de tesis doctoral es explorar cómo podemos encontrar estrategias de trading que muestren un rendimiento similar al obtenido en fase de optimización cuando sean evaluadas en nuevos datos, así como soluciones que puedan resistir cambios abruptos de tendencia y periodos extremos de volatilidad.

Proponemos extender un enfoque de promediado existente (Branke, 1998; Chen and Sundararaj, 1999; Branke, 2000; Jin and Sendhoff, 2003; Deb and Gupta, 2006) para encontrar soluciones robustas en problemas de uno o varios objetivos. En los problemas con un solo objetivo, como en el capítulo 6, utilizamos la idea de la función efectiva media, pero con una diferencia distintiva: en lugar de usar la vecindad de los parametros de la estrategia, evaluamos la función de fitness en muestras aleatorias del conjunto de datos de entrenamiento y calculamos un valor robusto de fitness haciendo el promedio del fitness obtenido en cada muestra aleatoria.

UNIVERSIDAD DE MÁLAGA

Nuestro enfoque multiobjetivo en el Capítulo 5 se basará en (Chen and Sundararaj, 1999) que sugirieron reemplazar el fitness f de la optimización convencional por dos objetivos, la media μ y la desviación típica σ de f.

II.III Trading Algoritmico

Los sistemas AT intentan capturar anomalías momentáneas en los precios, aprovechar patrones estadísticos dentro o entre los mercados financieros, ejecutar órdenes de forma óptima, ocultar las intenciones de un trader o detectar y explotar las estrategias de los rivales. La característica distintiva de los sistemas de negociación algorítmica es la sofisticación de su análisis y toma de decisiones (Nuti et al., 2011)

Una de las ventajas más importantes del trading algorítmico es que las estrategias se pueden probar en datos históricos. Esta capacidad de simular una estrategia es uno de los mayores beneficios del trading algorítmico. *Backtesting* nos dice como se hubiera comportado la estrategia en el pasado. Si bien el rendimiento comprobado no garantiza resultados futuros, puede ser muy útil para evaluar estrategias potenciales. Los resultados comprobados previamente pueden usarse para filtrar estrategias que no se ajustan al estilo de inversión requerido o que no es probable que cumplan los objetivos de rendimiento de riesgo / rendimiento.

Las metodologías de análisis financiero que empleamos en esta tesis son el análisis cuantitativo y técnico. El análisis cuantitativo ha dominado la industria financiera en las últimas décadas y es la base para la teoría de cartera moderna, la fijación de precios de derivados y la gestión de riesgos. El análisis cuantitativo trata los precios de los activos como aleatorios y utiliza un análisis matemático y estadístico para encontrar un modelo adecuado para describir esta aleatoriedad (Nuti et al., 2011).

El análisis técnico intenta pronosticar el movimiento futuro de los instrumentos financieros analizando los datos históricos de las cotizaciones, como el precio y el volumen. El análisis técnico se centra en los gráficos de movimiento de precios y en diversas herramientas analíticas para evaluar la fortaleza o debilidad de un activo y prever futuros cambios de precio. El análisis técnico ha sido parte de la práctica financiera durante muchas décadas, pero esta disciplina no ha recibido el mismo nivel de escrutinio y aceptación académica que los enfoques más tradicionales, como el análisis cuantitativo y fundamental. Sin embargo, varios estudios académicos sugieren que, a pesar de su jerga y métodos, el análisis técnico puede ser un medio eficaz para extraer información útil de los precios de mercado (Lo et al., 2000).

El análisis técnico se basa en la Teoría Dow, formulada por Charles H. Dow en una serie de editoriales de Wall Street Journal de 1900 a 1902. Estos editoriales explicaban las creencias de Dow sobre cómo se comportaba el mercado de valores. Dow creía que el mercado en su conjunto era una medida

UNIVERSIDAD DE MÁLAGA

confiable de las condiciones generales económicas y que al analizar todo el mercado, se podían determinar con precisión esas condiciones e identificar la dirección de las principales tendencias del mercado y las acciones individuales

Los siguientes dos supuestos básicos de la teoría de Dow son la base de todo el análisis técnico:

1. El mercado descuenta todos los factores que pueden afectar el precio de un valor.

2. Los movimientos de precios no son puramente aleatorios y se mueven en pautas y tendencias identificables que se repiten con el tiempo.

La suposición de que los mercados descuentan todos los factores, esencialmente quiere decir que el precio de mercado de un activo en un momento determinado refleja con exactitud toda la información disponible y por tanto, representa el verdadero valor razonable del activo. Esta hipótesis se basa en la idea de que el precio de mercado siempre refleja la suma total de conocimiento de todos los participantes en el mercado. La segunda hipótesis básica subyacente al análisis técnico, la noción de que los cambios de precios no son aleatorios y los precios se mueven en tendencias, tanto a corto como a largo plazo, que pueden identificarse. Una tendencia una vez se ha formado, tiende a persistir. Seguir la tendencia es una de las estrategias más típicas de traders técnicos.

El análisis técnico se usa para pronosticar el movimiento de precios de prácticamente cualquier instrumento financiero negociable que esté generalmente sujeto a fuerzas de oferta y demanda, incluyendo acciones, bonos, futuros y pares de divisas. De hecho, el análisis técnico puede ser visto simplemente como el estudio de las fuerzas de la oferta y la demanda, como se refleja en los movimientos de los precios de mercado de un valor.

En este trabajo de tesis doctoral hacemos uso del análisis técnico y cuantitativo como base de las estrategias que optimizaremos mediante usando un GA y GP.

III Problemas Abordados

En la segunda parte de la tesis tratamos dos problemas en el ámbito del AT con objecto de validar nuestras hipótesis de investigación.

III.I Optimización Robusta de TTSs prediseñadas

Las TTSs se definen como reglas de compra y venta derivadas del análisis técnico que a menudo implican diferentes parámetros que deben determinarse, los cuales influyen muchísimo en la calidad de las señales de compra y venta generadas. Uno de los problemas más comunes al optimizar los parámetros de una TTS es el sobre-ajuste. Durante la fase de optimización

UNIVERSIDAD DE MÁLAGA

se encuentran muy buenas soluciones para el conjunto de datos de entrenamiento, pero la mayoría de estas soluciones obtienen un rendimiento malo cuando se prueban con el conjunto de datos de evaluación. Este problema surge como consecuencia de seleccionar los parámetros que obtienen mayor rendimiento sin tener en cuenta su robustez y estabilidad, siendo por tanto muy frágiles a pequeñas perturbaciones. Las condiciones de los mercados son dinámicas y evolucionan continuamente, y cuando los parámetros obtenidos tras realizar la optimización se aplican al conjunto de datos de evaluación, cualquier pequeña perturbación en los datos produce una gran degradación del rendimiento.

La robustez se incorpora mediante la búsqueda de los parámetros de una TTS en múltiples mercados en lugar de un solo mercado. Utilizamos tanto la media como la desviación típica obtenida como objetivos robustos a optimizar. Las principales aportaciones pueden ser resumidas de la siguiente forma:

Adoptamos un enfoque robusto y multiobjetivo que utiliza un GA y que consiste en optimizar los mismos parámetros sobre múltiples índices bursátiles de diversos mercados, aumentando así las diferentes condiciones donde la solución propuesta tiene que funcionar. Nuestro método de optimización es capaz de producir un conjunto único de parámetros robustos que no sólo funciona bien en los diferentes mercados para los datos de la muestra, sino que también tiene un nivel similar o mejor de rendimiento cuando se prueban con datos nuevos. Los mercados tienen peculiaridades específicas, pero también comparten muchas características comunes. Aumentar el conjunto de datos para incluir varios índices, expone los patrones recurrentes presentes en los diversos mercados y ayuda a guiar al GA hacia ellos. Por otra parte, la optimización de un solo instrumento financiero dirige inexorablemente al algoritmo hacia los rasgos singulares de ese único mercado, reduciendo así la exposición a las diferentes condiciones variables de mercado a la que estaría expuesto el algoritmo de búsqueda.

Para validar nuestro enfoque, realizamos 4 experimentos diferentes:

1. La optimización del ratio de Sortino en cada mercado independientemente.

2. El anterior experimento es transformado en un problema multi-objetivo robusto. En vez de optimizar cada índice independientemente, se optimiza una TTS capaz de operar en todos. La función objetiva (ratio de Sortino) se transforma en dos objetivos, la media y la desviación típica del ratio de Sortino obtenido en todos los índices.

3. Optimización bi-objetivo del ratio de Sortino y la Perdida Máxima en cada mercado de forma independiente.

4. Volvemos a transformar el anterior experimento en una optimización

UNIVERSIDAD DE MÁLAGA

robusta multi-objetivo. Las dos funciones objetivo anteriores se transforman en 4 usando la media y la desviación típica del ratio de Sortino y la Perdida Máxima, obtenida en todos los mercados

La mayoría de las soluciones encontradas en todos los experimentos batieron sustancialmente a la estrategia *Buy & Hold* durante la evaluación. Las excepciones son los índices Dow Jones Industrial Average, Nasdaq 100 y S&P 500. Estos índices tenían mercados alcistas excepcionales donde es muy difícil de superar *Buy & Hold*. La interesante excepción fue el índice Nasdaq Composite, que ofreció resultados excepcionales en el experimento 1 casi duplicando el rendimiento de *Buy & Hold*, mientras que los experimentos 3 y 4 estaban ligeramente por debajo.

Los mejores resultados de evaluación se obtuvieron en el IBEX35 español y en el MIBTEL italiano, que se encontraban en mercados bajistas agudos durante la evaluación. La rentabilidad obtenida casi triplicó *Buy & Hold* ($-10\,\%$). Los retornos porcentuales son consistentes con los ratios de Sortino obtenidos. Ratios más altos de Sortino conllevan retornos mas altos mientras que ratios negativos de Sortino implican un retorno negativo.

Nuestra propuesta de metodología robusta ha mejorado los resultados durante la evaluación de la mayoría de los índices considerados en este capítulo, como lo demuestran las correlaciones más altas y el menor "encogimiento" de soluciones optimizadas en múltiples mercados. Nuestro trabajo ofrece evidencia de que la optimización sobre una amplia variedad de instrumentos financieros similares puede ayudar a producir soluciones robustas cuyo desempeño no se degrada significativamente cuando se prueban fuera de muestra o las condiciones del mercado cambian. Nuestro método robusto propuesto ha producido significativamente más soluciones cuyo rendimiento está cerca de la relación 1 : 1 óptima entre el conjunto de datos de entrenamiento y de evaluación que las ofrecidas por una optimización de mercado único. También ofrecemos pruebas de que la incorporación de la "Perdida Máxima" como un objetivo, conduce a mejores soluciones, ya que el GA tiene una manera de ver más detalles sobre el comportamiento de una estrategia, dando lugar a estrategias que el obtienen mejor rendimiento con menor riesgo.

III.II Diseño y Optimización Robusta de TTSs

En esta capítulo planeamos explorar cómo desarrollar desde "cero" una TTS robusta utilizando GP, que se empleará para gestionar una cartera de acciones del mercado español. El método investigado es aplicado para determinar las condiciones potenciales de compra y venta de las acciones, con el objetivo de producir soluciones robustas capaces de soportar condiciones extremas de mercado, produciendo al mismo tiempo altos rendimientos con un riesgo mínimo.

UNIVERSIDAD DE MÁLAGA

Abordamos de nuevo al problema de sobre-ajuste, quizás incluso agravado como consecuencia del uso de GP, ya que el tamaño del espacio de búsqueda aumenta en comparación con el último problema. Para resolverlo, exploramos un método de muestreo aleatorio que llamamos RSFGP. Nuestro método, en vez de evaluar el fitness de un individuo en todo el conjunto de datos de entrenamiento, selecciona n muestras tomadas al azar $\{s_1, s_2, \ldots, s_n\} \in \mathcal{S}$ de todo el conjunto de entrenamiento $\mathcal{S}$ y promedia el fitness obtenido en cada muestra. Esto se hace a nivel de individuo, es decir, cada individuo es evaluado en muestras diferentes durante la optimización.

Para validar nuestro método realizamos tres experimentos:

1. Utilizamos SGP (*Standard Genetic Programming*) sin ningún mecanismo de robustez para establecer una comparación de base.

2. Utilizamos el VAFGP (*Volatility Adjusted Fitness Genetic Programming*) propuesto por Yan and Clack (2010) como comparación de método robusto.

3. Presentamos nuestro metodo RSFGP que calcula el fitness mediante muestras aleatorias y lo comparamos con los dos anteriores.

Nuestro método RSFGP muestra mayor robustez y mejor rendimiento en los resultados sobre el conjunto de datos de evaluación en comparación con los métodos SGP y VAFGP.

El objetivo es diseñar soluciones que no solo obtengan un resultado similar durante el entrenamiento en la evaluación, sino que también puedan resistir cambios bruscos de tendencia y períodos extremos de volatilidad. Nuestro enfoque es sustancialmente diferente a trabajos previos en la literatura y se centra en cómo calcular la función de fitness usando un método de muestreo aleatorio.

El método propuesto es probado con una cartera de las 21 acciones de mayor capitalización y liquidez del mercado español utilizando 13 años de datos de precios diarios y se comparan los resultados obtenidos con el índice de mercado IBEX35. Los resultados obtenidos superan claramente a Buy & Hold, SGP y VAFGP. Las soluciones obtenidas demuestran claramente su robustez en el conjunto de datos de evaluación a las fuertes caídas del mercado, como se vio durante la crisis de deuda soberana europea experimentada recientemente en España. Las estrategias aprendidas fueron capaces de operar durante periodos prolongados, lo que demuestra su eficacia y robustez.

En resumen, el método desarrollado en este capítulo es capaz de desarrollar TTSs adecuadas para todas las condiciones de mercado con resultados prometedores, lo que sugiere un gran potencial en las capacidades de generalización del método propuesto. El uso de métricas financieras adicionales junto con indicadores técnicos permite que el sistema aumente el rendimiento mientras demuestra su resistencia a través del tiempo. El sistema RSFGP

UNIVERSIDAD DE MÁLAGA

es capaz de hacer frente a diferentes condiciones de mercado logrando una rentabilidad de la cartera del 31,81 % para el periodo de pruebas 2009 a 2013 en el mercado español, mientras el índice IBEX35 obtuvo un 2,67 % durante el mismo período.

IV Conclusiones

Las conclusiones principales que se pueden extraer de esta tesis es que hay margen de mejora en los enfoques actuales de optimización y búsqueda de estrategias de trading aplicando técnicas de optimización robusta evolutiva. Motivada la aceptación que las técnicas de computación evolutivas combinadas con la optimización robusta son capaces de gestionar la incertidumbre presente en las series temporales financieras generando estrategias de trading que baten significativamente al mercado.

En resumen podemos concluir lo siguiente:

- Las técnicas estándares de GA y GP son guiadas hacia picos de fitness muy alto produciendo estrategias de trading que obtienen rendimientos muy buenos durante el entrenamiento, pero muy pobre cuándo se comprueban con nuevos datos fuera de muestra.
- Para resolver el problema de sobre-ajuste se necesita un mecanismo de robustez que dirija al algoritmo hacia zonas del espacio de búsqueda donde el fitness es más alto de media.
- El problema se puede enfocar como un problema mono-objetivo donde la función objetivo original se promedia tomando muestras del conjunto de datos original a nivel del individuo. Es decir, cada individuo es evaluado en muestras aleatorias diferentes tomadas del conjunto de datos.
- También se puede enfocar como un problema multi-objetivo. La función objetiva original es reemplazada por la media y desviación típica de la función objetivo original.
- La robustez también se puede incorporar al optimizar conjuntamente una cesta de índices o acciones. En vez de generar una estrategia diferente para cada instrumento, se genera una estrategia única capaz de operar en todos ellos, lo que conlleva a soluciones más robustas.
- La robustez se puede medir calculando el "encogimiento" o el cambio en porcentaje de rendimiento entre el conjunto de entrenamiento y el de evaluación.
- Las soluciones robustas obtienen rendimientos satisfactorios cuando se evalúan en nuevos datos. Tienen muy poco "encogimiento" por lo

UNIVERSIDAD DE MÁLAGA

que ofrecen un rendimiento similar entre la fase de entrenamiento y evaluación.

- Los mecanismos de robustez expuestos son capaces de generar estrategias de trading capaces de operar durante prolongados períodos de tiempo sin requerir que el sistema se vuelva a optimizar.
- Las estrategias de trading robustas generadas son capaces de lidiar con situaciones extremas de mercado.
- Incluir la *Perdida Máxima* como un objetivo adicional de optimización de un problema multi-objetivo, o como el divisor en el ratio Sterling en un problema mono-objetivo genera soluciones con un perfil de riesgo menor que solamente utilizando el ratio de Sharpe o de Sortino.

V Trabajo Futuro

Debido al amplio rango de temas tratados en esta tesis, las futuras investigaciones se podrían concentrar en los siguientes asuntos:

- Por el lado algorítmico, realizar un análisis en profundiadad de los diferentes tipos de algoritmos podría ser de gran interés.
- Es bien sabido que la optimalidad de Pareto no obtiene buenos resultados con el número de objetivos es mayor de tres. Diferentes enfoques como el MOEA/D o IBEA se podrían considerar.
- Ese trabajo puede ser extendido con diferentes datos de otros mercados para poder continuar estudiando los efectos beneficiosos de los métodos multi-mercado de muestreo aleatorio.
- Extender el enfoque del problema incluyendo una cartera de acciones donde están permitidas las operaciones a largo y corto y donde las órdenes de compra como las ventas son coevolucionadas.
- Otras investigaciones podrían incluir una mayor variedad de indicadores técnicos y cuantitativos como la correlación de retornos y diferentes métricas de riesgo como el VaR (Valor en Riesgo) o el CVaR (Valor en Riesgo Condicional) en vez de la perdida máxima o la volatilidad.
- Los mercados financieros están sujetos a condiciones cambiantes en el tiempo, por lo que poder optimizar en un entorno dinámico podría ser muy interesante ya que los cambios pueden afectar a la instancia del problema, a las funciones objetivas y/o a las restricciones. Las técnicas de optimización dinámica podrían ser empleadas para seguir a la estrategia óptima de trading según varia en el tiempo.

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD DE MÁLAGA

Acknowledgements

I would like to thank my family, specially my wife, for her help and encouragement throughout these years. Without her help, this Ph.D. thesis would have not been possible.

I would also like express my gratitude to my thesis advisors Dr. Francisco Luna Valero and Dr. Francisco López Valverde for their guidance during the course of this research. This thesis would have not been possible without their effort. I would also like to give special thanks to Dr. David Quintana for his comments, suggestions and insightful discussions.

UNIVERSIDAD DE MÁLAGA

A mi familia

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD DE MÁLAGA

Artificial intelligence is
no match for natural stupidity
Albert Einstein

UNIVERSIDAD
DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA

UNIVERSIDAD DE MÁLAGA

Contents

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA

List of Figures

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD DE MÁLAGA

List of Tables

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA

Chapter 1

Introduction

In the business world,
the rearview mirror is always clearer
than the windshield.

Warren Buffett

1.1 Summary

Financial markets are one of the hardest environments where to apply Artificial Intelligence or Soft Computing techniques due to their inherent dynamic and stochastic nature. Financial time series are non-stationary and noisy and in most cases highly multi-dimensional rendering naive optimization methods only able to solve but the simplest problems.

EC (*Evolutionary Computation*), on the other hand, has shown that it is a very suitable algorithmic framework for solving financial engineering problems. EC can attribute its success to its population based search approach, as well as, its robustness to noisy fitness landscapes. This is reflected on the growing use of EC techniques by trading houses (Iba and Aranha, 2012).

EC is a family of search and optimization methods which computationally simulate the natural evolutionary process where survival of the fittest defines the success an individual and ultimately its ability to reproduce and transmit its genetic information to the next generation.

One of the greatest challenges of EC is coping with the uncertainty of financial markets. Trading strategies evolved with EC techniques tend to suffer from over-fitting. While good results are obtained during the optimization process, when these strategies are tested on new unseen data, they tend to perform very poorly. This thesis focuses on how we can obtain solutions that perform adequately when tested on new unseen data. We use a GA in chapter 5 and GP in chapter 6 for solving two classical financial problems, TTS (*Technical Trading Strategy*) optimization and TTS discovery

UNIVERSIDAD
DE MÁLAGA

(Aguilar-Rivera et al., 2015; Hu et al., 2015).

We deal with the problem of over-fitting from a robust optimization point of view, incorporating robust optimization methods that serve to guide the algorithms away from peaks of high fitness and towards areas of the search space where fitness is higher on average.

1.2 Research problem and hypothesis

The main problem this research tries to tackle is that of producting robust TTSs. Standard EC techniques are so good at tackling multimodal problems that they are guided towards peaks of high fitness. This produces solutions that perform well for the in-sample training data, but very poorly when tested out-of-sample.

Amongst other things, this research will try to answer the following questions:

- Can we use robust optimization techniques to produce TTS that perform well when tested on new unseen data?
- Does incorporating other time series from different financial instruments to the optimization process increase the robustness of solutions?

1.3 Research contribution

The overall goal of this thesis is to investigate robust optimization techniques using GAs (*Genetic Algorithms*) and GP (*Genetic Programming*) for technical trading strategy optimization and discovery. We aim to achieve this through incorporating robust optimization methods into the canonical GA and GP framework. Whilst progressing towards this objective, several research contributions are made.

The following list summarizes the main areas of novelty that this research makes innovations in, along with where they are first introduced in this document:

1.3.1 Contributions in Chapter 5

In chapter 5, we introduce a novel robust multi-market optimization approach and some other novel contributions which are detailed next:

- Robustness is incorporated by searching for the parameters of a TTS over multiple markets instead of just the single market of interest. We use both the mean and standard deviation obtained as robust objectives to be optimized.

UNIVERSIDAD DE MÁLAGA

- Multiple markets are used to expose the genetic algorithm to noise and variability in the environmental conditions. This generates robust solutions that produce similar performance when tested out-of-sample.

- Demonstrated the usefulness of including maximum drawdown as an additional optimization criteria.

1.3.2 Contributions in Chapter 6

In chapter 6, we introduce a RSFGP (*Random Sampling Fitness Genetic Programming*) method. This method finds robust strategies by optimizing over randomly sampled segments of the original dataset and using the mean value obtained in the samples as the fitness of the individual. The main characteristics of this novel contribution are:

- The use of a random sampling mechanism to divide the time series of a basket of stocks into segments without requiring user intervention, or any unsupervised machine learning method to groups segments into the distinct market conditions (bull, bear and sideways markets).

- The use of a robust fitness function that uses all sampled segments to calculate an overall fitness score across random market conditions reducing over-fitting of solutions.

- The use of different financial metrics never used together with technical indicators in previous research such as the returns, the moving average of returns, the CAPM (*Capital Asset Pricing Model*) alpha and beta, the Sharpe ratio and the volatility of the stocks calculated over different period lengths. (See table 6.2).

1.4 Publications

The following publications have been published during the course of this Ph.D. thesis:

- Berutich, J. M., Luna, F. and López, F. On the quest for robust technical trading strategies using multi-objective optimization. *AI Communications*, volume 27(4), pages 453–471, 2014.

- Berutich, J. M., López, F., Luna, F. and Quintana, D. Robust technical trading strategies using GP for algorithmic portfolio selection. *Expert Systems with Applications*, volume 46, pages 307–315, 2016.

UNIVERSIDAD DE MÁLAGA

1.5 Delimitations of scope

Limited effort has been made in thesis to examine the consequences of applying the methods investigated here in the real world. The focus of the work concentrates more on evolutionary computation and robust optimization than the issues relating to the practical implementation of the trading systems developed.

Figure 1.1 shows the focus of this thesis, which is at the intersection between algorithmic trading, roubst optimization and evolutionary computation.

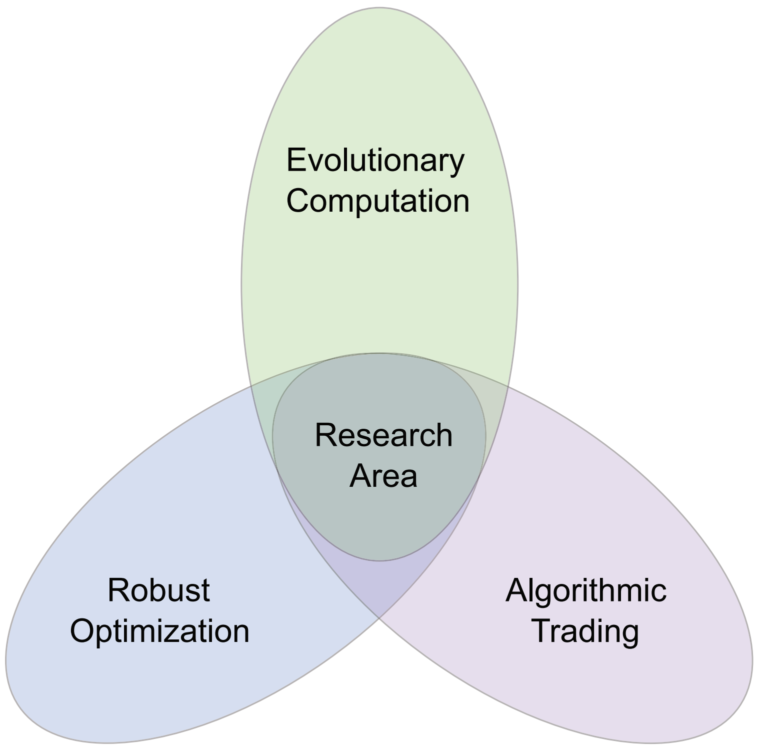


Figure 1.1: Scope of research conducted.

1.6 Document organization

This thesis is organized in the following manner:

- **Chapter 1 - Introduction**
  This chapter introduces the subject of this research, contributions, scope and publications supporting this thesis.

- **Chapter 2 - Metaheuristics**

UNIVERSIDAD DE MÁLAGA

We present the state of the art in metaheuristics for both single and multi-objective optimization and explore in more detail GA and GP which will be used throughout this thesis.

- **Chapter 3 - Robust evolutionary optimization**
This chapter introduces the state of the art in robust optimization in the context of evolutionary algorithms and how the most significant concepts extracted from the literature can be incorporated to both single and multi-objective optimization problems.

- **Chapter 4 - Algorithmic trading**
This chapter introduces the subject of algorithmic trading, its different components and the financial analysis methodologies used for the TTSs developed in this thesis. It analyses in more detail *Technical Analysis* and shows the technical indicators and some other calculations derived from *Quantitative Analysis* used during the course of this thesis.

- **Chapter 5 - Robust optimization of technical trading strategies**
This chapter presents a robust multi-market optimization methodology for TTS where robustness is incorporated via the environmental variables of the problem. The search for the optimum parameters is conducted over several markets (instead of just one). This exposes the GA to different market conditions and increases the robustness of the solutions generated. First, we review the most significant previous works on the subject, then we formally define the optimization problem solved, and continue by explaining the algorithmic approach employed. We also discuss the methodology used in our experimentation and the results obtained analysing the most relevant results, which highlight the strengths of our proposed methodology.

- **Chapter 6 - Robust technical trading strategy discovery**
This chapter extends on the work of the previous chapter, but this time using GP instead to discover robust TTSs that are used to manage a portfolio of stocks from the Spanish stock market. Firstly we introduce the subject and continue by discussing the most relevant recent research in the area. Then the problem is described together with the algorithmic approach used and methodology employed for the experimentation conducted testing our research questions.

- **Chapter 7 - Conclusions and Future work**
This chapter presents the most important conclusions that have been extracted during the course of this research work and the proposed lines of future work could focus on.

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA

Part I

State of the art

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA

Chapter 2

Metaheuristics

God created not species,
but the algorithm for creating species
(Paraphrased)

Charles Babbage

2.1 General optimization background

Optimization can be easily described as the process of finding the best solution to a specified problem, or in more mathematical terms, finding the maximum or minimum values of a function subject to certain constrains. Optimization is an area of great importance in computer science, mathematics, engineering and science in general as any type of problem can be posed as an optimization problem.

In order to develop an understanding of optimization some formal non-ambiguous definitions are needed:

Definition 1 (Single-objective optimization) *A single objective optimization problem is defined as minimizing (or maximizing) $f(x)$, subject to $g_i(x) \leq 0$, $i = \{1, ..., m\}$, and $h_j(x) = 0$, $j = \{1, ..., p\}$, $x \in \Omega$. A solution minimizes (or maximizes) $f(x)$ where x is a n-dimensional decision variable vector $x = (x_1, ..., x_n)$ from some universe Ω.*

Note that $g_i(x) \leq 0$ and $h_j(x) = 0$ represent the constraints that must be met while maximizing or minimizing $f(x)$. Ω contains all possible x that can be used to satisfy an evaluation of $f(x)$ and its constraints. x can be a vector of continuous or discrete variables as well as f being continuous or discrete. The method for finding the global optimum (which may not be unique) of any function is referred to as Global Optimization. In general, the global minimum of a single objective problem is presented in Definition 2.

UNIVERSIDAD DE MÁLAGA

Definition 2 (Single-Objective Global Minimum) *Given a function* $f:\Omega \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$, $\Omega \neq 0$, *for* $x^* \in \Omega$ *the value* $f^* \triangleq f(x^*) \geq -\infty$ *is called a global minimum if and only if:*

$$\forall x \in \Omega : f(x^*) \leq f(x) \tag{2.1}$$

$\mathbf{x}^*$ *is by definition the global minimum solution,* f *is the objective function, and the set* Ω *is the feasible region of* $\mathbf{x}$*. The goal of determining the global minimum solution(s) is called the* ***global optimization problem*** *for a single-objective problem.*

The problems defined above are approached with algorithms that try to find the global optimum. These methods can be classified into two categories: *exact*, and *heuristic*. Exact methods guarantee the optimal solution will be found for a given optimization problem, and they are the preferred method if they can solve an optimization problem with an effort that grows polynomially with the problem size. On the other hand, problems that are $\mathcal{NP}$-hard (Garey and Johnson, 1979) need exponential effort and even medium-sized problems often become intractable and can not be solved any more using this methods. In order to overcome such limitations heuristic methods can be applied. Heuristics can not guarantee that an optimal solution will be found, but can provide acceptable approximate (also optimal) solutions in a reasonable time (Blum and Roli, 2003; Rothlauf, 2011). Among the basic heuristic methods we can distinguish between *constructive* and *local search* methods. Constructive algorithms generate solutions by adding to an originally empty partial solution components until a solution is complete. On the other hand, local search methods start from an initial solution and iteratively try to improve it, with a better solution in a defined neighbourhood of the current solution (Blum and Roli, 2003). Figure 2.1 shows an overview of the different optimization approaches.

Some examples of exact optimization methods are the *Enumerative*, *Branch and bound*, *Depth-first*, *Breadth-first*, *A** and *Z**. Lastly, *calculus*-based methods require continuity in some variable domain for an optimal value to be found (Cormen et al., 2009; Rothlauf, 2011).

Even though the exact methods mentioned above have all been applied successfully to solve a wide variety of problems (Cormen et al., 2009), when problems are high-dimensional, discontinuous, multi-modal and/or $\mathcal{NP}$-hard, exact methods are ineffective. On the other hand, heuristic search and optimization methods were developed to address the limitations of exact search and optimization algorithms. Heuristic methods require a function that measures the quality of solutions, and a mapping mechanism to encode/decode between the problem and algorithm domains. In general, they provide effective solutions to a wide variety of optimization problems which conventional exact search methods find unmanageable (Blum and Roli,

UNIVERSIDAD DE MÁLAGA

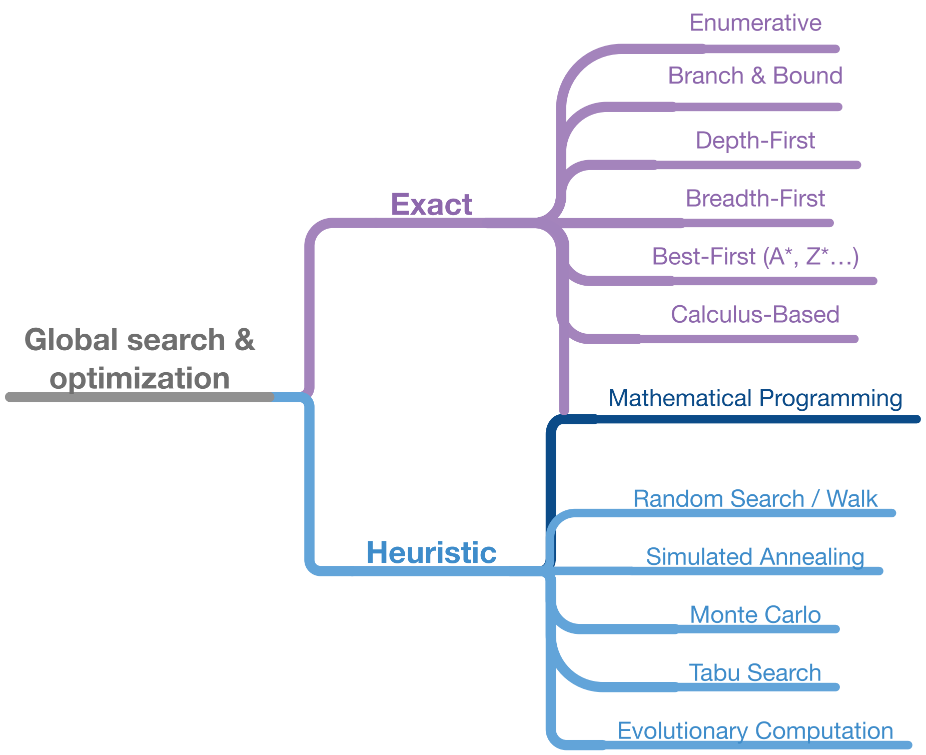


Figure 2.1: Global optimization approaches

2003).

The simplest heuristic search strategy is *random search*. It simply evaluates a given number of randomly selected solutions. A *random walk* is very similar, except that the next solution is randomly selected using the last evaluated solution as a starting point. Like *enumeration*, these strategies are not effective for many problems because they do not incorporate domain knowledge. Random searches can generally expect to do no better than enumerative ones. *Monte Carlo* methods simulate stochastic events employing a random search and saving the best solution for comparison purposes.

Methods like *Simulated Annealing*, *Tabu search* and *Evolutionary Computation* (among others), are also known as *metaheuristics* are discussed next.

UNIVERSIDAD DE MÁLAGA

2.2 Metaheuristics

"*Metaheuristics*, in their original definition, are solution methods that orchestrate an interaction between local improvement procedures and higher level strategies to create a process capable of escaping from local optima and performing a robust search of a solution space" (Glover and Kochenberger, 2003; Gendreau and Potvin, 2010). The term was first introduced by (Glover, 1986) and fuses the greek prefix *meta* (higher, beyond) with *heuristic* (from the Greek heuriskein or euriskein, to discover, to find out).

Metaheuristics can be applied to a large variety of optimization problems, as very few assumptions about the problem are made (Sörensen and Glover, 2013). Normally they are used in problems when it is not possible to implement the optimal method or where no other algorithm or heuristic is capable of producing adequate solutions. Most metaheuristics are non-deterministic and implement some form of stochastic optimization.

Compared to exact algorithms, metaheuristics do not guarantee that a globally optimal solution can be found on some classes of problems (Blum and Roli, 2003), but can provide near optimal solutions for most classes of problems and do offer some important advantages over exact methods as metaheuristics do not require that the function may be differentiable or continuous and incorporate mechanisms for escaping from local optima. Metaheuristics can be considered useful approaches for solving complex optimization problems as they can often find good solutions with less computational effort than exact or iterative methods (Blum and Roli, 2003; Bianchi et al., 2009; Gendreau and Potvin, 2010; Lones, 2014).

The fundamental characteristics of metaheuristics can be summarized in the following outline:

- Metaheuristics are strategies to guide the search process.
- The objective is to explore the search space efficiently in order to find (near)-optimal solutions.
- Metaheuristics are problem-independent.
- Metaheuristics are approximate and non-deterministic algorithms.
- Metaheuristics range from simple local-search to complex learning processes.
- Metaheuristics incorporate mechanisms to escape from local optima.

Metaheuristics can be classified into many different categories according to what distinguishing criterion is chosen (see Figure 2.2). According to

UNIVERSIDAD DE MÁLAGA

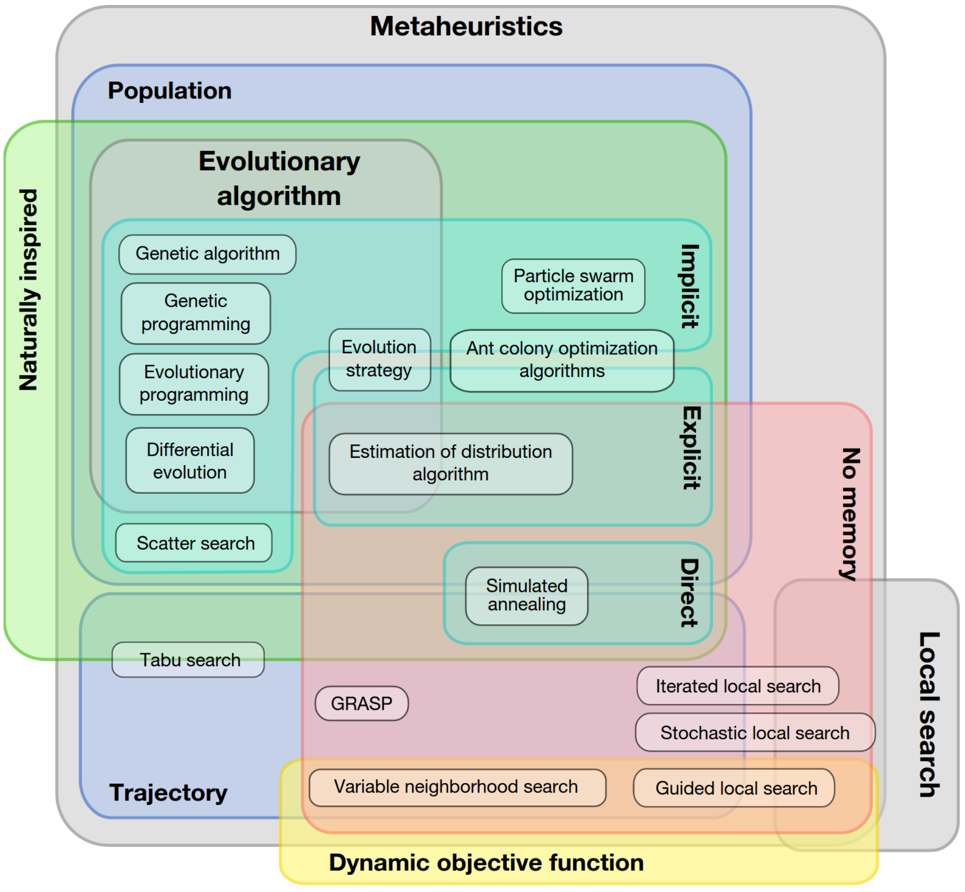


Figure 2.2: Metaheuristics classification, adopted from (Wikipedia, 2016)

Blum and Roli (2003) the most important ways of classifying metaheuristics are:

- *Trajectory vs. population-based*: *Trajectory* based metaheuristics iteratively improve a single solution. Some famous examples are *Tabu Search* and *Simulated Annealing*. On the other hand, *population* based metaheuristics improve a set of candidate solutions using a population based approach. Some examples of population based metaheuristics are *Particle Swarm Optimization*, *Ant Colony Optimization* and *Evolutionary Algorithms*.

- *Nature-inspired* vs. *non-nature inspired*: Some examples of metaheuristics based on *nature-inspired* processes are *Evolutionary Algorithms*, *Ant Colony Optimization*, *Particle Swarm Optimization*, versus *non-nature inspired* like *Iterated Local Search*, *GRASP* or *Tabu Search*.

UNIVERSIDAD DE MÁLAGA

- *Dynamic vs. static* objective function: While some metaheuristics keep the objective function static during the optimization process, others like *Guided Local Search* modify the objective function by incorporating information collected during the search process, the idea behind this approach is to escape from local optima by modifying the search landscape. *Variable Neighborhood Search* uses a set of neighborhood structures which gives the possibility to diversify the search by swapping between different fitness landscapes.

- *Memory* usage vs. *memory-less* methods: Another very important way of classifying metaheuristics according to whether they use the search history or not. The use of memory nowadays is recognized as one of the fundamental elements of a powerful metaheuristic. Some examples of *memory-less* methods are *Simulated Annealing* and *Guided Local Search*.

This thesis deals specifically with EAs (*Evolutionary Algorithms*) . An EA is based upon Darwinian principles of evolution, where a population of multiple candidate solutions is evolved using biologically inspired processes such as reproduction, mutation, recombination and selection.

Algorithm 1 Template of an EA

```
1: $P(0) \leftarrow$ GenerateInitialPopulation()
2: $t \leftarrow 0$
3: Evaluate($P(0)$)
4: while not StoppingCriterion( ) do
5:     $P'(t) \leftarrow$ Selection($P(t)$)
6:     $P''(t) \leftarrow$ Reproduction($P'(t)$)
7:     Evaluate($P''(t)$)
8:     $P(t+1) \leftarrow$ Replacement($P(t)$, $P''(t)$)
9:     $t \leftarrow t+1$
10: end while
```

A typical EA follows the pseudo-code included in Algorithm 1. In EAs, candidate solutions are called *individuals*. Groups of individuals are referred to as *populations*. As in real life, some individuals from a population mate (i.e., are selected) for reproduction, generating new children or offspring which, according to the natural selection process, may replace other members of the population.

Whenever a new solution is created, it is evaluated to be assigned a fitness, which is an indicator of the quality of the solution in the context of the problem being solved. When the current population is replaced by a new one, a generation has taken place. The process of iterating through successive generations is called evolution, and ends when a termination condition is fulfilled.

UNIVERSIDAD DE MÁLAGA

EAs make no assumption about the underlying fitness landscape, this generality makes them capable of approximating solutions to all types of problems. EAs have demonstrated its success in fields as diverse as biology, economics, engineering, finance, logistics, machine learning, operations research, physics, robotics and social sciences (among others) (Haupt and Haupt, 2004; O'Reilly et al., 2005; Chen et al., 2007; Poli and Langdon, 2007; Jaimes and Coello, 2008; Maulik et al., 2011; Iba and Aranha, 2012; Brabazon et al., 2012; Khouadjia et al., 2013; Mutingi and Mbohwa, 2017).

Some popular EAs are GA (*Genetic Algorithm*), GP (*Genetic Programming*), ES (*Evolution Strategy*) and EP (*Evolutionary Programming*). This thesis deals specifically with two types of evolutionary algorithms, GA and GP, which will be presented next in sections 2.2.1, and 2.2.2 respectively.

2.2.1 Genetic algorithms

Genetic algorithms were pioneered by (Holland, 1975) and were popularized by his former Ph.D. student (Goldberg, 1989). GAs are robust search and optimization techniques inspired by natural evolution and genetics. GAs are very effective in problems were the search space is very large, complex and multimodal. *Chromosomes* encode solutions on a string-like data structure, resembling the DNA code of living organisms.

A GA typically starts with a randomly generated population of individuals or chromosomes. Each individual is then evaluated, and a fitness measure is obtained which determines the quality of the solutions encoded as chromosomes. Individuals are then probabilistically selected for reproduction. During selection, individuals with higher fitness are given a better chance to reproduce than others members of the population which may have a worse fitness value. *Crossover* and *mutation* are then applied on the selected individuals. The crossover operator mixes the chromosomes of the parent individuals to form new offspring, whereas mutation alters some part of the individual's chromosome. This process goes on for a number of generations or until some stopping criterion is met.

2.2.1.1 Characteristics

Genetic Algorithms differ from most traditional optimization techniques in several ways. GAs encode the decision variables of the problem in the chromosome (genotype) where the genetic material gets manipulated during crossover and mutation, and not directly on the variables themselves (phenotype).

GAs are regarded as blind search techniques that explore via sampling using only pay-off information and employ stochastic operators to guide the search and be able to escape from local optima. GAs are inherently parallel as they can evaluate simultaneously a set of encoded solutions or individuals.

UNIVERSIDAD DE MÁLAGA

Another important capability of GAs is their ability to successfully explore search spaces which may not be continuous nor differentiable.

Normally GAs perform best when solutions can be represented or encoded in a manner which exposes important components of possible solutions, and operators to mutate and hybridize these components are available. On the other hand, GAs are hindered when the selected encoding does not fully describe the main characteristics of potential solutions, or when operators are not able to generate interesting new candidate solutions. The following list provides a summary of the main characteristics of GAs:

- The decision variables are represented as fixed length strings (*chromosomes*).
- Inherently parallel *population* based approach. Multiple *individuals* are evolved at the same time.
- The *fitness function* determines the quality of potential solutions.
- Biologically inspired operators are used for selection/reproduction and mutation.
- Genetic operators are applied probabilistically to a fraction of the population.
- Evolution continues until some stopping criterion is met.

2.2.1.2 Solution encoding and initialization

The first thing that needs to be addressed in order to solve an optimization problem using GAs, is the encoding of the decision variables into chromosomes. As explained earlier, chromosomes are represented as a fixed length string over an alphabet of fixed length (Holland, 1975; Goldberg, 1989). A commonly used principle for encoding is known as the principle of minimum alphabet (Holland, 1975). The principle states that for efficient encoding, the smallest alphabet set that permits a natural expression of the problem should be chosen. The most simple and common representation is a binary string. A binary string of length l can encode 2^l solutions.

For example the string [1 0 1 0 1 0 1 0] of length 8 can encode a maximum of 256 (2^8) solutions, this is what is defined as the *genotype*. On the other hand, the *phenotype* is the manifestation of the genotype, (i.e. the actual observed properties) and represents the solution to our problem. Going back to our 8-bit string example, the 8-bit string could represent 8 different items we wish to include in a box in a packing problem where we need to maximize the sum of the weight of the items without reaching a specified maximum weight.

UNIVERSIDAD DE MÁLAGA

There is a mapping needed from the genotype to the phenotype. In this simple example each position or gene in the chromosome represents a predefined item in our problem. The allele is the value that the gene can take. In our example problem it can have two values, 1 means in the box or 0 not in the box.

The binary alphabet offers the maximum number of schemata per bit of information of any encoding (Holland, 1975; Goldberg, 1989) and is one of the most commonly used strategies. Other types of encoding such as integer and real value encoding are also widely employed. Ultimately the encoding depends on the nature of the problem being optimized and the data type of the decision variables. It is also common practice to mix both integer and real values in the chromosome, as some of the decision variables of the problem might be either integer or real, as is the case with our trading system optimization problem of chapter 5.

The set of all chromosomes or individuals is defined as the population, the size of which may be fixed or may vary on every generation. It is general practice to choose the initial population randomly. However, one can generate the chromosomes of the initial population using some domain knowledge about the problem. For example a regularly used technique is to “seed” or introduce some good known solutions in the initial population.

2.2.1.3 Fitness evaluation

The next step is to evaluate the fitness of the encoded solutions. The fitness function is problem-dependent and measures the quality of a solution. The fitness function should be selected in such a way that a chromosome that is nearer to the optimal solution in the search space receives a higher (or lower in case of minimization) fitness value. The fitness function (also called the payoff information or utility function) is the only information that GAs require to explore the search space.

In single-objective optimization only one fitness value is used for optimization, whereas in multi-objective optimization, a different fitness value is calculated for each of the objective functions optimized. We will discuss this in more detail in section 2.3.2.

2.2.1.4 Selection

The selection operator simulates the process of natural selection. The success of an individual is determined by “survival of the fittest” concept of Darwinian evolution theory. The selection procedure consists in generating an intermediate population, called *mating pool*, by copying the chromosomes from the parent population. Several widely used techniques for selection are *tournament selection* and *roulette wheel*. In tournament selection, two (or more) chromosomes are taken at random, and the best chromosome is put

UNIVERSIDAD DE MÁLAGA

into the mating pool. The process is repeated until the mating pool becomes full. Tournament selection may be implemented with and without replacement of the competing individuals. The tournament size also can be varied. It should be noted that in tournament selection, the worst chromosome will never be selected in the mating pool.

2.2.1.5 Crossover

Crossover is a genetic operator that combines two parent individuals to produce new offspring and is based on the idea of sexual reproduction where offspring may be better than both parents if they inherit their best characteristics. Crossover is normally applied during evolution according to a user-defined probability. Most crossover operators are designed "ad-hoc" customizing them to the problem being solved. Some popular crossover operators are single-point, two-point, n-point and uniform crossover. Crossover operators are designed depending on the specific problem at hand. For example combinatorial optimization problems like the *Travelling Salesman Problem* may use specific permutation operators such as *ordered crossover*, *partially mapped crossover* or *edge recombination crossover*.

2.2.1.6 Mutation

The mutation operator modifies one or more genes in a chromosome with the goal of preventing the population from stagnation at local optima. Mutation is applied during evolution according to a user-specified probability which is normally set quite low (somewhere between 1% and 10% is a good starting point). A very high probability of applying mutation can turn the search into a random search. Another important application of the mutation operator is repairing solutions which may violate some constrains.

2.2.1.7 Elitism

Elitism guarantees that the best chromosome found so far is not lost due to randomized operators during the search. In elitist models, the best chromosome seen up to the current generation is retained either within the population itself or in an outside location. Sometimes elitism is performed by replacing the worst individual of the current generation by the best of the previous generation (only if it has better fitness).

2.2.2 Genetic programming

First introduced by (Koza, 1992), "GP (*Genetic Programming*) is an EC (*Evolutionary Computation*) technique for automatically solving problems without requiring the user to know or specify the form of the solution in

UNIVERSIDAD DE MÁLAGA

advance. At the most abstract level GP is a *systematic*, *domain-independent* method for getting computers to solve problems *automatically* starting from a high-level statement of what needs to be done (Poli and Langdon, 2007)". Since its inception twenty five years ago, GP has been succesfully employed to solve a wide variety of practical problems, producing a number of human-competitive results and patentable new inventions (Poli and Langdon, 2007; Riolo et al., 2010, 2016).

2.2.2.1 Representation

GP is similar to a GA but has significant differences in the way solutions are represented. Whereas a GA employs a fixed-length string encoding, GP employs a variable length representation in the form of *syntax* trees. Figure 2.3 shows the tree representation of a TTS (*Technical Trading Strategy*) `and (`$\texttt{EMA}_{20}$` > `$\texttt{EMA}_{50}$`,`$\texttt{RSI}_{14}$` < 30)`. The variables and constants in the program, *the terminals*, ($\texttt{EMA}_{20}$, $\texttt{EMA}_{50}$, $\texttt{RSI}_{14}$ and 30) are leaves of the tree. The operators (`<,>,and`) are internal nodes called *functions*. Functions and terminals form the *primitive set*.

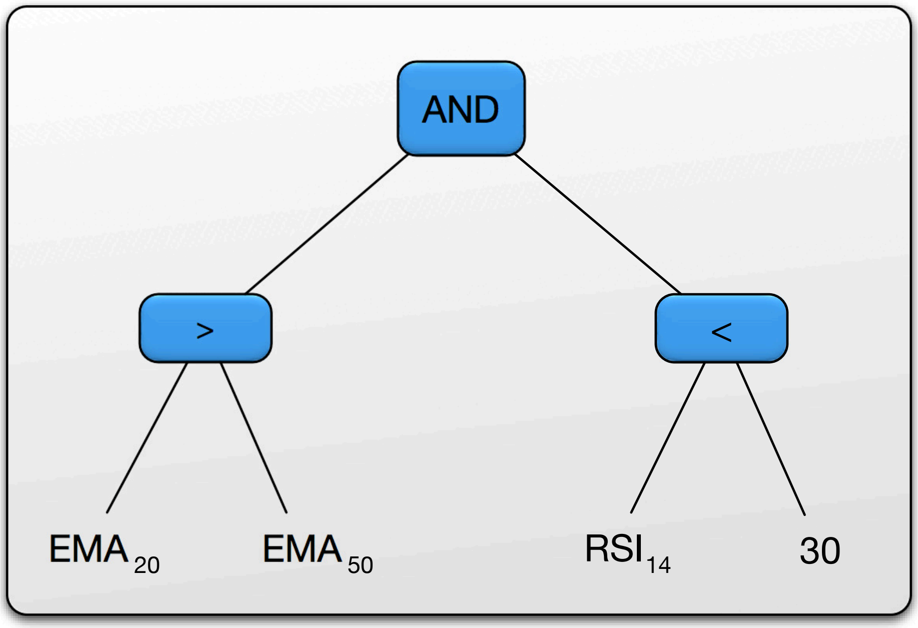


Figure 2.3: GP syntax tree representing the *Technical Trading Strategy* `and (`$\texttt{EMA}_{20}$` > `$\texttt{EMA}_{50}$`,`$\texttt{RSI}_{14}$` < 30)`

However, it is common in most GP applications for *functions* to have a fixed number of arguments, hence trees can be effectively represented as simple lists or linear sequences. From the name of the function the arity

UNIVERSIDAD DE MÁLAGA

is determined and the brackets can be inferred. For example the previous expression in prefix-notation `(and (> EMA`$_{20}$ `EMA`$_{50}$`) (< RSI`$_{14}$ `30))` could be written unambiguously as the sequence `and > EMA`$_{20}$ `EMA`$_{50}$ `< RSI`$_{14}$ `30`. The choice of using either an explicit tree representation or a linear sequence is typically dictated by convenience, efficiency, the genetic operators used and the data that needs to be collected during the optimization run (Poli and Langdon, 2007).

2.2.2.2 Population initialization

As in other EA (*Evolutionary Algorithms*)s, in GP the individuals in the initial population are randomly generated. We will describe next two of the simplest approaches for initializing the population, the `full` and `grow` methods together with a widely used combination of both known as `ramped half-and-half` (Koza, 1992).

In both full and grow methods, the initial population is generated without exceeding a user specified maximum *depth*. The depth of a node is determined by the number of edges that need to be traversed in order to arrive to it, and the depth of the tree is the depth of its deepest leave. The root node is assummed at depth 0. The `full` method produces trees where all leaves are at the same depth, this does not mean that the trees will have an identical shape or size (the number of nodes in the tree). In fact, this only happens when all the functions in the primitive set have the same arity. Anyhow, even with mixed-arity primitive sets, the range of program sizes and shapes produced by the full method tends to be very limited. On the other hand, the `grow` method produces trees with a greater diversity of sizes and shapes. The grow method selects both functions and terminals from the primitive set until the maximum depth is reached. Once the maximum depth is reached only terminals are selected. The `grow` method is very sensitive to the sizes of the function and terminal sets. For example, if there are significantly more terminals than functions, the `grow` method will almost always produce very short tress irrespective of the depth limit. Similarly, if the number of functions is considerably greater than the terminals, then the `grow` method will behave much like the `full` method (Poli and Langdon, 2007).

Because both the `full` and `grow` methods are unable to produce a wide variety of sizes and shapes by themselves, Koza (1992) proposed a combination of both called `ramped half-and-half`. In `ramped half-and-half`, half of the population is generated using the `full` method while the remaining other half is created with the `grow` method. This is performed using a range of depth limits ("ramped") to guarantee that the population has a diversity of different sizes and shapes.

The initialization procedure need not be always random. As in other EAs domain knowledge can be incorporated by initializing the population with

UNIVERSIDAD DE MÁLAGA

likely properties of desired solutions or previously known good solutions.

2.2.2.3 Selection

In GP *selection* is performed in very much the same way as in GAs. The selection operators described in Section 2.2.1.4 are also some typical selection approaches applied in GP.

2.2.2.4 Crossover

GP significantly differs from other EAs in how it implements the crossover. The most common crossover is the subtree crossover or single-point crossover. Given two parents, the single-point crossover randomly and independently chooses a node as crossover point in each of the parents. It then creates the offspring by replacing the subtree rooted at the crossover point of the first parent by the subtree at the crossover point of the second parent. Figure 2.4 shows this operator. This is also the crossover operator employed in the experiments of Chapter 6. Note that is possible to return two offspring but this is not common. (Poli and Langdon, 2007).

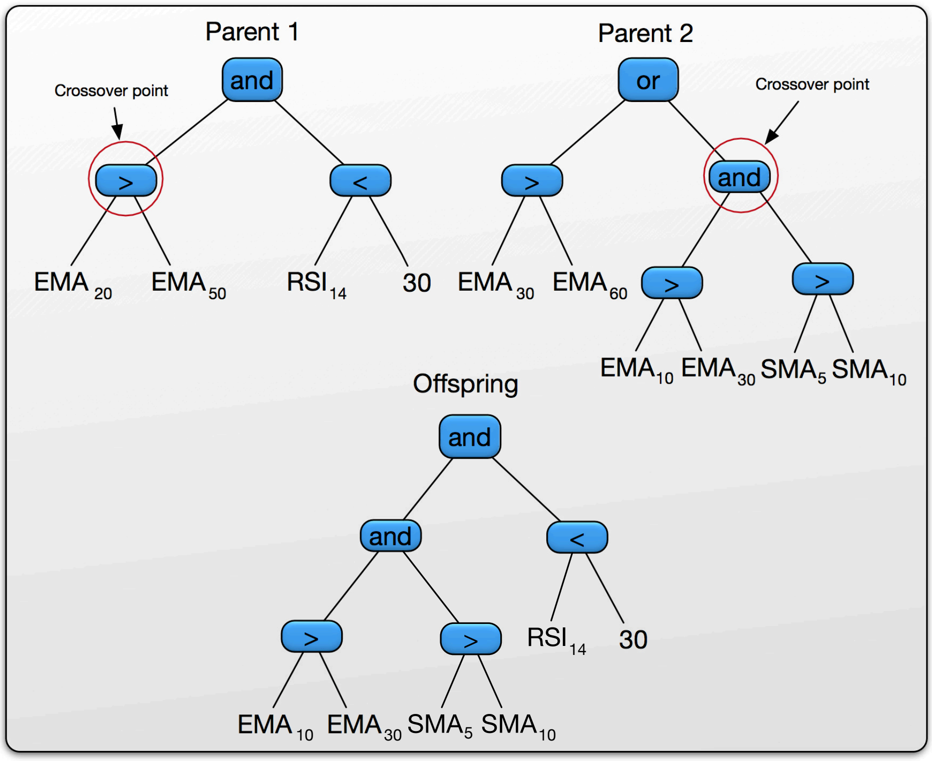


Figure 2.4: GP subtree or single-point crossover

UNIVERSIDAD DE MÁLAGA

"Often crossover points are not selected with uniform probability. Typical GP primitive sets lead to trees with an average branching factor (the number of children of each node) of at least two, so the majority of the nodes will be leaves. Consequently the uniform selection of crossover points leads to crossover operations frequently exchanging only very small amounts of genetic material (i.e., small subtrees); many crossovers may in fact reduce to simply swapping two leaves. To counter this, (Koza, 1992) suggested the widely used approach of choosing functions 90% of the time and leaves 10% of the time" (Poli and Langdon, 2007).

2.2.2.5 Mutation

The implementation of the mutation operator is also quite different from other EAs. The most common form of mutation in GP is the *uniform* mutation or *subtree* mutation, where a point in the tree is randomly chosen and the subtree rooted there is exchanged for another randomly generated subtree. This is the type of mutation that we will use in Chapter 6.

Another common mutation operator is *point mutation*, which is similar to GAs *bit-flip mutation*. In point mutation a random point is chosen and the primitive stored there is replaced with a random primitive of the same arity. If no other primitives with the same arity exist, no mutation is applied to that node (but other nodes might still be mutated).

Operators are applied probabilistically and in GP they are normally applied in a mutually exclusive way, unlike other EAs where offspring are generated often applying a combination of crossover and mutation. Typically crossover is applied with the highest probability often at 90% or higher. Mutation is applied with a much lower probability usually in the region of 1% to 10%. When crossover or mutation do not get applied to an individual, the individual simply gets copied to the next generation.

2.2.2.6 Primitive set

GP can be seen as a technique that can evolve programs, not in the Turing-complete languages typically used for software development, but in a more constrained, domain-specific language specified by the Primitive set. Primitives can be either *functions* or *terminals* and are explained next.

2.2.2.7 Terminal set

The terminals are the leafs of the tree and consist of:

- The *program's external inputs* (i.e. the named variables of the problem `EMA`, `RSI`, ...,).

UNIVERSIDAD DE MÁLAGA

- *Functions with no arguments* like the function `random()` which returns a random number or the function `altitude()` which returns the distance to the ground of a drone GP is flying.
- *Constants* which can be pre-specified or randomly generated when creating the tree.

Using a terminal such as the function `random()` will cause the behavior of the individual program to vary every time it is executed, even when it is called with the same inputs. This might be desirable in some applications, however in most cases we need a set of fixed random constants that are generated during the population initialization. This can be accomplished by including an *ephemeral random constant*. Every time this type of terminal is chosen during population initialization or during sub-tree mutation a different random value is generated which is used in that specific terminal and remains fixed for the rest of the run.

2.2.2.8 Function set

The function set is normally dictated by the nature of the problem being solved. For example in simple symbolic regression problems the functions may consist of the arithmetic functions (`+`, `-`, `*`, `/`) and may be extended with boolean function (`AND`, `OR`, `NOT`), mathematical functions (`log`, `exp`, `cos`, ...).

All sorts of other functions and constructs found in computer programs can be employed. Most times, the primitive set includes specialized functions and terminals designed to solve problems in a specific domain. For example if we are programming a drone to fly we might include such actions as `take-off`, `turn`, `forward` and `land`.

2.2.2.9 Closure

"For GP to work effectively, most function sets are required to have an important property known as closure (Koza, 1992), which can in turn be broken down into the properties of *type consistency* and *evaluation safety*." (Poli and Langdon, 2007)

Type consistency requires that all functions return values of the same type and that their arguments also have the same type, as crossover operations can mix and join nodes arbitrarily. This limitation can be worked-around by using *strongly-typed GP*. In strongly-typed GP the functions and terminals are defined specifying its arguments and return data types. Special crossover and mutation operators explicitly use this information so that the offspring they produce do not contain illegal type mismatches.

Evaluation safety is required because many commonly used functions can fail at run time. For example an expression might divide by 0. This is

UNIVERSIDAD DE MÁLAGA

typically dealt with by using protected versions that first test for potential problems with its inputs before execution. If a problem is detected some default value is returned. For example protected division checks if the second argument is 0 and if so returns the first arguments if that is the case, otherwise it performs the division.

2.2.2.10 Sufficiency

Primitive sets must be sufficient for GP to produce satisfactory results. "Sufficiency means it is possible to express a solution to the problem at hand using the elements of the primitive set. Unfortunately, sufficiency can be guaranteed only for those problems where theory, or experience with other methods, tells us that a solution can be obtained by combining the elements of the primitive set" (Poli and Langdon, 2007).

If a primitive set is insufficient, GP is only able to generate programs that approximate the desired one. However, such an approximation can be very near and good enough for the decision maker. Adding a few unnecessary primitives in order to ensure sufficiency does not tend to slow down GP much, although there are cases where it can bias the systems in unexpected ways (Poli and Langdon, 2007).

2.2.2.11 Fitness function

The primitive set defines the search space GP will explore, however we still are not able to determine which elements or regions of the search space constitute good solutions that can solve or approximately solve the problem at hand. As discussed earlier, the fitness function measures the quality of individuals. The main difference of the fitness function in GP compared to other EAs is that in GP programs need to be interpreted to be executed. Interpreting a program tree means executing the nodes in an order that guarantees that no nodes are executed before the value of their arguments. This is usually accomplished by traversing the tree recursively starting from the root node and delaying the evaluation of each node until the values of its arguments are known. Other orders like traversing the tree from leaves to the root is also possible if there are no functions which produce side effects.

2.3 Multi-objective optimization

There are many real-word problems that have opposing objectives which need to be optimized simultaneously (as is our case in the problem presented in Chapter 5). This can be tackled using multiobjective optimization algorithms, which present a set of solutions representing trade-offs between the different objectives.

UNIVERSIDAD DE MÁLAGA

2.3.1 Formal definition

A general MOP (*Multiobjective Optimization Problem*) can be formally defined as follows (we assume minimization without loss of generality).

Definition 3 (MOP) *Find a vector $\vec{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)$ which satisfies the m inequality constraints $g_i(\vec{x}) \geq 0, i = 1, 2, \ldots, m$, the p equality constraints $h_i(\vec{x}) = 0, i = 1, 2, \ldots, p$, and minimizes the vector function $\vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x}))$, where $\vec{x} = (x_1, x_2, \ldots, x_n)$ is the vector of decision variables.*

The set of all values satisfying the constraints defines the *feasible region* Ω and any point $\vec{x} \in \Omega$ is a *feasible solution.*

We seek for the *Pareto Optimal Set.* Before its definition some concepts must be introduced.

Definition 4 (Pareto Optimality) *A point $\vec{x}^* \in \Omega$ is Pareto optimal if for every $\vec{x} \in \Omega$ and $I = \{1, 2, \ldots, k\}$ either $\forall_{i \in I} f_i(\vec{x}) = f_i(\vec{x}^*)$ or there is at least one $i \in I$ such that $f_i(\vec{x}) > f_i(\vec{x}^*)$.*

This definition states that $\vec{x}^*$ is Pareto optimal if no feasible vector $\vec{x}$ exists which would improve some criterion without causing a simultaneous worsening in at least another criterion. Other important definitions associated with Pareto optimality are the following:

Definition 5 (Pareto Dominance) *A vector $\vec{u} = (u_1, \ldots, u_k)$ is said to dominate $\vec{v} = (v_1, \ldots, v_k)$ (denoted by $\vec{u} \preccurlyeq \vec{v}$) if and only if $\vec{u}$ is partially smaller than $\vec{v}$, i.e., $\forall i \in I,\ u_i \leq v_i \ \wedge\ \exists i \in I :\ u_i < v_i$.*

Definition 6 (Pareto Optimal Set) *For a given MOP $\vec{f}(\vec{x})$, the Pareto optimal set is defined as $\mathcal{P}^* = \{\vec{x} \in \Omega | \neg \exists \vec{x}' \in \Omega, \vec{f}(\vec{x}') \preccurlyeq \vec{f}(\vec{x})\}$.*

Definition 7 (Pareto Front) *For a given MOP $\vec{f}(\vec{x})$ and its Pareto optimal set $\mathcal{P}^*$, the Pareto front is defined as $\mathcal{PF}^* = \{\vec{f}(\vec{x}) | \vec{x} \in \mathcal{P}^*\}$.*

Obtaining the Pareto front of a MOP is the main goal of multi-objective optimization. However, given that a Pareto front can contain a large number of points, a good solution can only include a limited number of them, which should be as close as possible to the true Pareto front (*convergence*), as well as being uniformly spread (*diversity*); otherwise, they would not be very useful to the decision maker. Closeness to the Pareto front, or convergence, ensures that we are dealing with (almost) optimal solutions, while a uniform spread of the solutions, or diversity, means that we have made a good exploration of the search space and no regions are left unexplored.

UNIVERSIDAD DE MÁLAGA

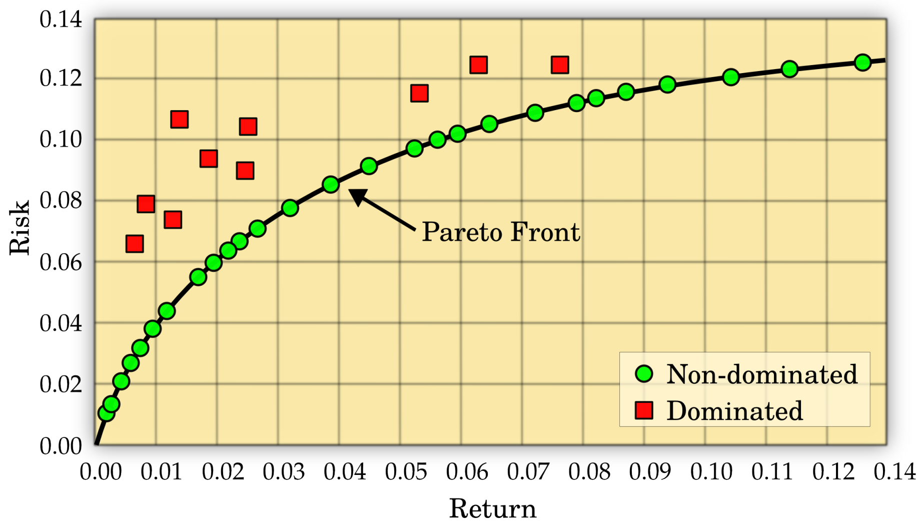


Figure 2.5: An example of a problem with two objective functions: risk (minimization) and return (maximization). The Pareto front or trade-off surface is delineated by a curved line.

The normal procedure to approximate the Pareto front is to generate as many points as possible in Ω, and then filter them out to keep those that are non-domintated. As MOPs usually have a possible uncountable set of solutions in their Pareto front, the approximation has to sample them properly to maintain a good distribution so as to cover all the regions, i.e., the different trade-off between the problem objectives (Coello et al., 2007). A sample Pareto front is shown in Figure 2.5 depicting dominated and non-dominated solutions.

2.3.2 Metaheuristics for multi-objective optimization

Traditional metaheuristics are single-objective by nature as they optimize a single criterion during the search process. Hence, a single solution is produced which corresponds to the best value for the chosen optimization criterion. However, most practical engineering optimization problems are multi-objective as they must consider simultaneously various performance criteria which are often conflicting. This can be handled in using *a priori*, *interactive* or *a posteriori* methods.

2.3.2.1 A priori techniques

This group of techniques assume that the decision maker is able to specify the desired goals or a certain pre-ordering of the objectives prior to carrying out the search. This is usually accomplished by aggregating the objectives

UNIVERSIDAD DE MÁLAGA

in a weighted manner. Essentially, the preferences of the decision maker are modeled so that solutions may be compared to them.

According to Coello et al. (2007), the most significant *A priori* approaches can be summarized as follows :

- **Lexicographic ordering:** The objectives are ranked by importance as defined by the decision maker and the optimum solution is obtained by minimizing the various objective functions sequentially in order of importance starting from the most important objective. Typically the objectives not being optimized are handled as constrains to the problem.

  It may also be possible to assign randomly the importance of the objectives when they are unknown. This might be similar to a weighted combination of the objectives where each weight is defined in terms of the probability of each objective being selected. The main weakness of this approach is that it tends to favour certain objectives while relegating others when many are present due to the randomness involved in the process. This technique seems suitable for problems where the importance of each objective in comparison to others is well known beforehand. This technique has not found much acceptance among multi-objective researches due to the fact that it explores the search space unevenly, as priority is given to solutions performing well in one objective but not in the others.

- **Linear aggregating functions:** This approach aggregates the functions in a weighted manner typically in the form:

$$\text{fitness} = \sum_{i=1}^{k} w_i f_i(x) \tag{2.2}$$

  where $w_i \geq 0$ and $i = 1, \ldots, k$ are the weight coefficients representing the relative importance of the objectives as specified by the decision maker.

- **Nonlinear aggregating functions:** Multiplicative approaches, where the objective functions are multiplied, are not very popular in the literature, since the definition of a good nonlinear aggregating function may result harder than defining a linear aggregating function. On the other hand, target-vector approaches, which require the definition of goals by the decision maker are more popular, specially in the case when the decision maker is able to specify a vector with the desired goals for each objective. Target-vector functions typically take the following

UNIVERSIDAD DE MÁLAGA

form:

$$\text{fitness} = \sum_{i=1}^{k} \left(\frac{f_i^0 - f_i(x)}{f_i^0} \right)^p \tag{2.3}$$

where f^0 is the vector of desired goals specified by the decision maker, and $i = 1, \ldots, k$ are the k objective functions in the MOP.

2.3.2.2 Interactive techniques

Interactive techniques integrate search and decision making. These methods typically work in three phases:

1. Find a non-dominated solution.
2. Receive feedback on the solution from the decision maker. Modify the preferences of the objectives accordingly.
3. Repeat previous steps until the decision maker is satisfied or no further improvement is possible.

Some well known interactive algorithms are the *Probabilistic Trade-Off Development* method, the *STEP* method and the *Sequential Multiobjective Problem Solving* method (Coello et al., 2007).

2.3.2.3 A posteriori techniques

A posteriori techniques do not require the input of the decision maker to conduct the search. The decision maker is only needed after the optimization process is finished to select the most appropriate solution that fulfills satisfactorily all the objectives of the problem. A posteriori techniques can be further classified into (Mashwani et al., 2016):

- **Pareto-based:** Pareto-based techniques make use of the concept of *Pareto Dominance* to establish preference between solutions. The aim is to obtain a set of non-dominated solutions representing a good approximation to the Pareto optimal front. While Pareto-based methods perform satisfactorily on MOPs with a limited number of objectives, for many-objective problems, Pareto dominance becomes a weak criteria for selecting solutions, as the number of feasible solutions that when compared are non-dominated becomes too large. This causes the algorithm to not have enough convergence pressure. Another limitation of Pareto-based methods is the number of solutions required to accurately approximate Pareto fronts, as it grows exponentially with

UNIVERSIDAD DE MÁLAGA

the number of objectives (Aguirre, 2013). To cope with these limitations for many-objective problems, decomposition and indicator based approaches have been proposed showing very good results (Li et al., 2013). Some of the most representative Pareto-based algorithms are: NSGA-II, SPEA2, PAES, PESA, PESA-II. This algorithms stand out because they incorporate known MOEA (*Multiobjective Evolutionary Algorithm*) theory (Coello et al., 2007).

- **Decomposition-based:** Decomposition techniques reduce the problem into smaller sub-problems and optimize them either sequentially or in parallel. MOEA/D introduced by Zhang and Li (2007) is perhaps the most well-known decomposition algorithmic framework. This framework incorporates a mathematical method to transform a multi-objective problem into several single-objective problems which are simultaneously solved. *Normal Boundary Intersection* and *Chebyshev* are some examples of the mathematical methods that may be incorporated into the MOEA/D framework.

- **Indicator-based:** Perhaps the most important trend in modern MOEAs (after 2002) is the use of a performance measure in the selection operator. Quality metrics such as the hypervolume that measure the portion of objective space dominated by a set of solutions, are used to direct the search in a Pareto-compliant way. The advantage of indicator-based metaheuristics is that their convergence pressure does not decrease as the number of objective are increased. The hypervolume is the only performance indicator that is known to be monotonic with respect to Pareto dominance. This guarantees that the true Pareto front achieves the maximum possible hypervolume value, and any other set will produce a lower value for this indicator. Some representative examples are IBEA (Indicator-Based Evolutionary Algorithm) proposed by Zitzler and Künzli (2004) is an algorithmic framework that allows the incorporation of any performance indicator in the selection mechanism of a MOEA. It was originally tested with the *hypervolume* and the binary ϵ indicator. Another important MOEA is SMS-EMOA proposed by Beume et al. (2007), which uses a steady state selection and the operators from NSGA-II. The hypervolume is used as a density estimator which improves the distribution of solutions along the Pareto front. The solution that is dominated by the largest number of solutions is removed.

2.4 Conclusions

In this chapter we have formally introduced an optimization problem, and discussed the state of the art in metaheuristics. We have explored in more de-

UNIVERSIDAD DE MÁLAGA

tail the two techniques (GA and GP) used in this thesis. Lastly, we formally define a MOP and the concepts of Pareto Optimality, Pareto Dominance, Pareto Optimal Set and Pareto Front and cover the approaches typically employed for solving MOPs.

UNIVERSIDAD DE MÁLAGA

Chapter 3

Robust evolutionary optimization

Fragility is the quality of things that are vulnerable to volatility.

Nassim Nicholas Taleb

3.1 Introduction

The term "*robust*" has many definitions depending on the author, but it can be broadly defined as the capacity of a system to preserve its functionality despite internal (genotypic robustness) or external perturbations (phenotypic robustness) (Branke, 1998; Soule, 2003). Real-world solutions to optimization problems should also be robust as their performance should be unaffected by small changes to the design variables or the environmental conditions. This is not usually considered in traditional optimization algorithms which assume these variables as constant. Robustness and performance can be conflicting and so it is important to know their relation for each optimization problem, therefore the goal of robust optimization techniques should be to not only maximize performance, but also to guarantee sufficient robustness (Gaspar-Cunha and Covas, 2008).

Robustness is usually measured after optimization conducting a sensitivity analysis that measures the response of the optimized solutions to small parameter variations. Since the solutions generated considering only performance, and not taking into account its variation (robustness), might not include the best results, a robustness analysis should be performed during the search (Beyer and Sendhoff, 2007; Gaspar-Cunha and Covas, 2008), which can be accomplished by introducing a measure of robustness during optimization that replaces the original objective function with an expression measuring both performance and robustness or by measuring expected

UNIVERSIDAD DE MÁLAGA

performance around the vicinity of a solution. It can also be approached as a multi-objective optimization problem where the trade-off between performance and robustness (usually conflicting objectives) is obtained. (Jin and Sendhoff, 2003; Deb and Gupta, 2005, 2006) considered the problem of finding robust solutions in single-objective optimization as a multi-objective optimization problem where robustness and performance are maximized.

3.2 Types of robustness

In this section we will attempt to classify the types of robustness according to whether robustness is internal to the evolutionary process (*genotypic* robustness) or external (*phenotypic* robustness) (Yan and Clack, 2010).

3.2.1 Genotypic robustness

Genotypic robustness aims at achieving insensitivity of fitness to perturbations from genetic operators. In Soule (2003) robustness refers to resistance to change from variation operators such as crossover and mutation and finds that the *code bloat* phenomenon in GP, where an increase of the size of the trees does not result in fitness improvement, is a redundancy mechanism. Trees grow *introns* which safeguard valuable code and protect it against loss during crossover or mutation. Even though this approach favors broad plateaus instead of peaks of high fitness, it is of negligible use when the surface of the search space changes. Their research shows that evolutionary pressure favoring robust solutions has a significant impact on the evolutionary process.

Piszcz and Soule (2006) study again the robustness to genetic operators in GP in the context of redundancy and replication of the building blocks in an individual. If growth of the GP solution is allowed, it permits the individual to support more disruptive events caused by crossover or mutation. Robustness decreases the probability that a critical subtree or building block will affect the overall fitness of the individual.

3.2.2 Phenotypic robustness

Phenotypic robustness deals with resilience to external changes and can be further categorized into:

- *Generalization Robustness:* Robustness as the generalization ability of evolved solutions. From a machine learning standpoint, it is the predictive accuracy of a learner for new unseen cases. The objective here is reducing over-fitting and producing solutions whose performance is similar for both in-sample and out-of-sample datasets (Kushchu, 2002).

UNIVERSIDAD DE MÁLAGA

- *Environmental Robustness:* Robustness to external environmental perturbations. Financial markets suffer abrupt structural changes which tend to persist in time (Granger and Hyung, 2004). Robust TTSs should withstand periods of extreme volatility and trend change.
- *Robustness to Noise:* Robustness to noise inherent in the data or the readings produced by the system (Kitano, 2004).
- *Self Repair:* Robustness as the ability to self-repair after severe phenotypic damage (Bowers, 2006).

In this thesis we are concerned with the *generalization* and *environmental* robustness of evolved solutions. We explore how we can design solutions that display similar performance for both in-sample and out-of-sample data, as well as solutions that can resist abrupt trend changes and extreme volatility periods.

3.3 Definitions

In the next section we introduce the mathematical formulation to robust optimization problems.

3.3.1 Robustness in single-objective optimization

Given a single-optimization problem of the following type:

$$\left.\begin{array}{rl} \text{Minimize} & f(x) \\ \text{subject to} & x \in \mathcal{S} \end{array}\right\} \tag{3.1}$$

where $\mathcal{S}$ is the feasible search space, a robust solution is defined as being insensitive (to a limit) to the perturbation of the design variables x.

Definition 8 (Single-objective robust solution) *In the minimization of function $f(x)$, a solution x^* is a robust solution, if it is the global-minimum of the mean effective function $f^{eff}(x)$ defined with respect to a δ-neighborhood as:*

$$\left.\begin{array}{ll} \textit{Minimize} & f^{eff}(x) = \dfrac{1}{|\mathcal{B}_\delta(x)|} \displaystyle\int_{y \in \mathcal{B}_\delta(x)} f(y)dy \\ \textit{subject to} & x \in \mathcal{S} \end{array}\right\} \tag{3.2}$$

where $\mathcal{B}_\delta(x)$ is the δ-neighborhood of solution x and $|\mathcal{B}_\delta(x)|$ is the hypervolume of the neighborhood.

UNIVERSIDAD DE MÁLAGA

Let us consider Figure 3.1, of two minimum solutions, solution A is considered robust as its objective function value is not altered significantly by a small perturbation of the decision variables. On the other hand, solution B is quite sensitive to the perturbation of the design variables and can not be recommended even though it has a lower function value than solution A.

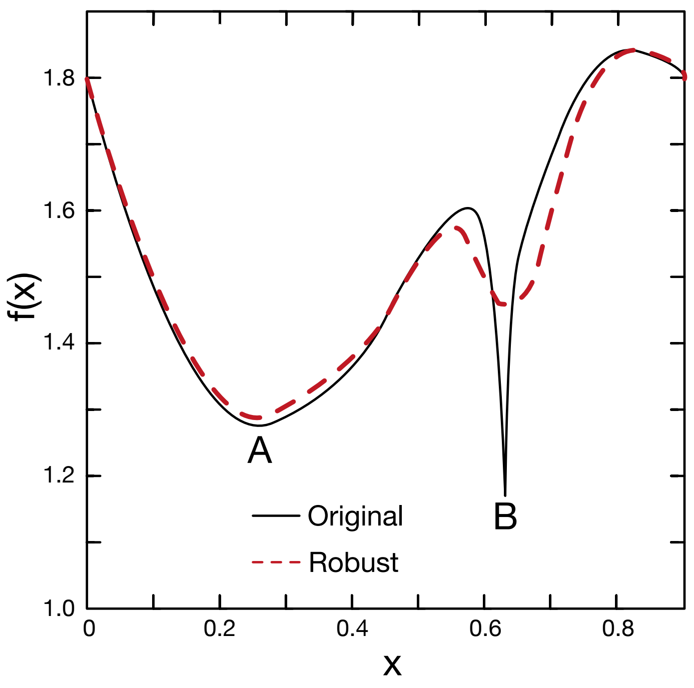


Figure 3.1: Global verus robust solutions in single-objective optimization

One of the main ideas that can be extracted from the literature is to use the mean effective objective function for optimization instead of the objective function itself. To estimate the expected performance the fitness of a solution (x) is calculated by averaging several points in its neighborhood as shown in Equation 3.3 (Tsutsui and Ghosh, 1997; Branke, 1998; Jin and Sendhoff, 2003; Deb and Gupta, 2006; Beyer and Sendhoff, 2007; Gaspar-Cunha and Covas, 2008).

$$\tilde{f}(x) = \frac{\sum_{i=1}^{N} w_i f(x + \Delta x_i)}{\sum_{i=1}^{N} w_i} \tag{3.3}$$

where x is the vector of design variables and possibly some environmental parameters, $i = 1, 2, \ldots, N$ is the number of points to be evaluated. Δx_i is a vector of small numbers that can be generated deterministically or randomly. w_i is the weight for each evaluation. In the simplest case all weights are set

UNIVERSIDAD DE MÁLAGA

equally to 1. If Δx_i are random variables z drawn according to a probability distribution $\phi(z)$, we obtain in the limit $N \to \infty$, the effective evaluation function f_{eff} according to the following equation 3.4 (Tsutsui and Ghosh, 1997; Jin and Sendhoff, 2003).

$$f_{eff} = \int_{-\infty}^{\infty} f(x, z)\phi(z)dz \tag{3.4}$$

Besides averaging methods, it has been shown by Tsutsui and Ghosh (1997) that the "*perturbation*" of design variables in each fitness evaluation leads to the maximization of the effective fitness function of Equation 3.4 under the assumption of linear selection. *Schema theorem* is used as a basis for the mathematical proof. Note that the "*perturbation*" method is equivalent to the averaging method for $N = 1$ and a stochastic Δx (Jin and Sendhoff, 2003).

We extend this concept in Chapter 6 by using randomly sampled subsets of the dataset in each evaluation of the individual and averaging the fitness obtained in each evaluation. Equation 3.5 shows this.

$$\bar{f}(x) = \frac{\sum_{i=1}^{N} f(x, z_i)}{N} \tag{3.5}$$

where $i = 1, 2, \ldots, N$ are the random samples to be evaluated. x are the design variables and z_i are random samples of the environmental variables of the problem (the dataset z). In our case as we will see in the following chapters, z_i are ramdom samples taken from the timeseries of prices of financial assets. What this method accomplishes is using randomly selected segments from the original timeseries as a method to embed robustness during single-objective optimization.

3.3.2 Robustness in multi-objective optimization

Given a multi-objective optimization problem with M conflicting objectives:

$$\left.\begin{array}{rl} \text{Minimize} & (f_1(x), f_2(x), \ldots, f_m(x)) \\ \text{subject to} & x \in \mathcal{S} \end{array}\right\} \tag{3.6}$$

where f_m are the M objective functions, x is the vector of decision variables and some environmental parameters and $\mathcal{S}$ is the feasible search space. In contrast to single-objective optimization, where the goal is finding a single optimum, in multi-objective optimization a finite number of Pareto-optimal solutions are searched for. The two main differences with respect to single-objective robust optimization can be enumerated as:

1. The sensitivity of the design variables x now has to be established with respect to all M objectives. A combined effect of the variations in all

UNIVERSIDAD DE MÁLAGA

M objectives has to be used as a measure of robustness. Alternatively, the decision maker may choose which are the preferred objectives that require insensitivity to variable perturbation.

2. In multi-objective optimization there are numerous solutions that need to be analyzed for robustness.

Multi-objective robust solutions can be defined as:

Definition 9 (Multi-objective robust solution) *A solution x^* is a multi-objective robust solution if it is the global-feasible Pareto-optimal solution to the following multi-objective problem defined with respect to a δ-neighborhood ($\mathcal{B}_\delta(x)$) of a solution x:*

$$\left.\begin{array}{ll} \textit{Minimize} & (f_1^{eff}(x), f_2^{eff}(x), \ldots, f_m^{eff}(x)) \\ \textit{subject to} & x \in \mathcal{S} \end{array}\right\} \tag{3.7}$$

where $f_j^{eff}(x)$ is defined as:

$$f_j^{eff}(x) = \frac{1}{|\mathcal{B}_\delta(x)|} \int_{y \in \mathcal{B}_\delta(x)} f_j(\mathbf{y}) d\mathbf{y} \tag{3.8}$$

Figure 3.2 shows two Pareto-optimal solutions (A and B) which are analyzed for their sensitivity in the decision variable space. Since the perturbation of point B causes a large degradation in the objective values, solution B does not qualify as a robust solution. In order to qualify as robust solution, each Pareto-optimal solution has to be insensitive to small perturbations in its decision variables. On the other hand, solution A is robust, as its objective function values show less variance certifying its insensitivity towards small perturbations in the decision variable space.

Solutions obtained with conventional evolutionary optimization algorithms will be the most performing, not the most robust. In order to incorporate robustness, for each objective function, an additional criterion needs to be introduced, which measures the variation of the original objective function around the vicinity of the design point considered. Variance measures only take into account deviations of the function, ignoring the associated performance. Thus, in the case of a single objective function the optimization algorithm must perform a two-criterion optimization (f_m,f^R), where f_m concerns performance and f^R accounts for robustness (Branke, 1998; Chen and Sundararaj, 1999; Branke, 2000; Gaspar-Cunha and Covas, 2008).

3.4 Robustness approach employed

In this thesis we are concerned with the *generalization* and *environmental* robustness of evolved solutions. We explore how we can design solutions that

UNIVERSIDAD DE MÁLAGA

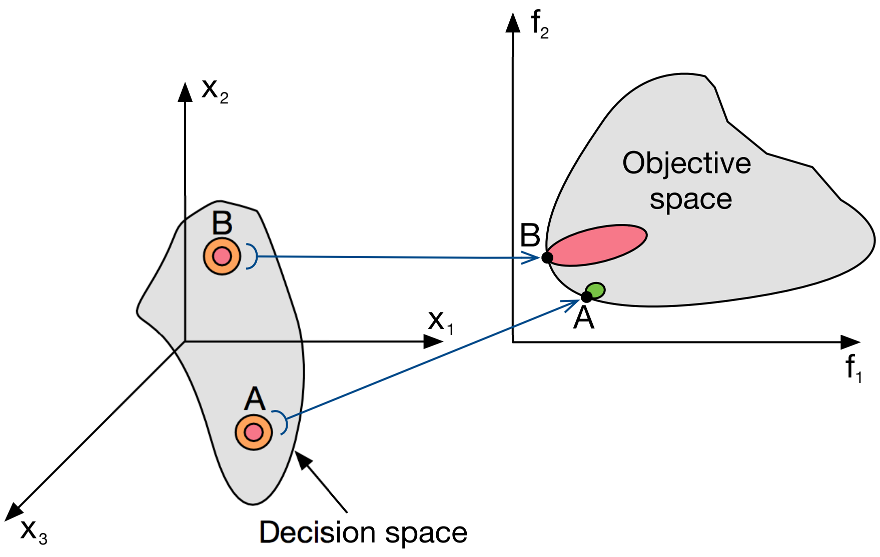


Figure 3.2: Point A is less sensitive than point B to variable perturbation in a multi-objective problem

display similar performance for both in-sample and out-of-sample data, as well as solutions that can resist abrupt trend changes and extreme volatility periods. We make an effort to extent an existing averaging approach (Branke, 1998; Chen and Sundararaj, 1999; Branke, 2000; Jin and Sendhoff, 2003; Deb and Gupta, 2006) for finding robust solutions in single-objective and multi-objective problems. In single-objective problems, as in Chapter 6, we use the idea of the mean effective function, but with a distinctive difference: instead of using the vicinity, we evaluate the fitness function on randomly sampled points from the original dataset and compute a robust fitness by averaging the fitness obtained on each random sample.

Our multi-objective approach in Chapter 5 will be based on (Chen and Sundararaj, 1999) who suggested replacing the fitness (f) of conventional optimization (Definition 1) by two objectives, the mean μ and the standard deviation σ of f.

3.5 Conclusions

In this chapter we have presented the state of the art in robust evolutionary optimization. The main concept that can be drawn from the literature is to use the mean effective fitness calculated in the vicinity of the parameters. The optimization problem can also be treated as a multi-objective problem, considering the statistical dispersion of the mean effective fitness as an addi-

UNIVERSIDAD DE MÁLAGA

tional objective. Using multi-objective optimization would require to double the number of optimization criteria as each original objective function needs to be substituted by the mean and standard deviation of the original fitness function measured in a vicinity of the parameters being optimized.

UNIVERSIDAD DE MÁLAGA

Chapter 4

Algorithmic trading

Ever wonder why
fund managers can't beat the S&P 500?
'Cause they're sheep,
and sheep get slaughtered.

Gordon Gekko (Wall Street 1987)

4.1 Introduction

AT using evolutionary computation has been a hot topic of research in the recent years for academics from both finance and soft-computing domains with a large body of published research articles (Ponsich et al., 2013; Hu et al., 2015; Aguilar-Rivera et al., 2015).

AT is a term commonly used to describe computer programs that automate one or more stages of the trading process. AT currently handle approximately 50% to 60% of all stocks traded in the United States and the European Union and is a major source for computing and analytics innovation, specially machine learning and grid/GPU computing (Nuti et al., 2011; Hendershott and Riordan, 2013).

AT systems try to seize momentary anomalies in market prices, profit from statistical patterns within or across financial markets, optimally execute orders, conceal a trader's intentions, or detect and exploit rivals' strategies. Ultimately, profits drive any algorithmic trading system whether in the form of cost savings, client commissions, or proprietary trading. The distinguishing characteristic of algorithmic trading systems is the sophistication of their analysis and decision making (Nuti et al., 2011).

These systems are deployed in highly liquid markets in assets classes such as equities, futures, derivatives, bonds, and foreign exchange. Algorithmic trading is employed for automating any stage of the trading process, therefore it covers a wide variety of systems. In trade-execution programs, for example,

UNIVERSIDAD DE MÁLAGA

the algorithm might decide aspects such as which market to send the order to, the timing, price, and even the order's quantity splits (Nuti et al., 2011).

One of the most important benefits of algorithmic trading is that strategies can be tested on historical data. This ability to simulate a strategy is one of the biggest benefits of algorithmic trading. Back-testing tells you how well the strategy would have done in the past. While back-tested performance does not guarantee future results, it can be very helpful when evaluating potential strategies. Back-tested results can be used to filter strategies that either do not suit the required investment style or are not likely to meet risk/return performance goals.

4.2 Financial time series

Financial time series are built at the most basic level by recording each (*tick*) where the *price*, *volume* and a *time-stamp* of each transaction are recorded asynchronously as it takes place. If we resample the tick data into periods of equal length, we have what is called OHLCV (*Open, High, Low, Close, Volume*) data. *Open* and *Close* represent the first and last prices of the interval, while *High* and *Low* prices are the maximum and minimum prices recorded during the interval. *Volume* represents the total number of financial instruments that where exchanged between sellers and buyers during the interval. We can assume that for any given market the price series is given by vectors $\{Open, High, Low, Close, Volume\}$ sampled at interval τ (minutes, hour, day, etc.).

4.3 Algorithmic trading system components

According to Nuti et al. (2011), the major components of an algorithmic trading system can be classified according to what stage of the trade's life cycle is being automated. The trading process can be split into four distinct stages: *pre-trade analysis*, *trading signal generation*, *trade execution*, and *post-trade analysis*.

- *Pre-trade analysis* involves analyzing financial data or news with the goal of forecasting future price movement or volatility and generating trading signals when an opportunity presents. Some methodologies used to perform this step are *fundamental analysis*, *technical analysis* and *quantitative analysis*.

- *Trading signal generation* and pre-trade analysis generally overlap blurring their differences. Their major distinction is that an actual trading signal generated by an algorithm will come with an specific price and quantity, and might also include risk management recommendations

UNIVERSIDAD DE MÁLAGA

such as specific stop-loss values. Normally pre-trade analysis recommendations are intentionally left vague, as they might be augmented by trading signal generation in more complex systems.

- *Trade execution* takes care of submitting the order to the trading venues. If the order is too large, the system breaks down the order into smaller orders which it submits over a period of time in order to minimize market impact. Orders might also be submitted to multiple markets, crossing markets or dark pools that do not publicly reveal the current order book.
- *Post-trade analysis* handles the analysis of the all transactions executed and measures the performance of the trading activity providing insights.

In this thesis we have fused both pre-trade analysis and trading signal generation as in our experiments we simulate a fixed quantity investment which we buy at the price when the signal is given.

4.4 Pre-trade analysis methodologies

Traditional financial forecasting methodologies can be classified into three categories, fundamental, quantitative and technical analysis. Next we will introduce briefly Fundamental analysis, and we will discuss in more depth Quantitative Analysis and Technical Analysis, specially the indicators used in this thesis.

4.4.1 Fundamental Analysis

Fundamental analysis studies the basic financial information of a company in order to forecast its profits, sales, costs, growth potential, management abilities, and other intrinsic matters affecting the market value of a stock (Thomsett, 1998). In fundamental analysis, investors estimate the value of a corporation using various financial valuation models. The data inputs for these models are derived from various accounting metrics appearing in the different financial statements and reports of a corporation such as assets, liabilities, earnings, etc. Investors try to find mispriced stocks trading in the market at a discount to their valuation models.

In this thesis we do not make use of fundamental analysis. The main reason is that the basic financial information is released to the public on a quaterly basis. The trading systems we simulate in our experiments operate at 30 minute frequency in Chapter 5 and daily frequency in Chapter 6. Including fundamental analysis into a trading system would make sense only on systems that trade on very long frequecies, for example in the case where stocks are held for periods longer than 6 months.

UNIVERSIDAD DE MÁLAGA

4.4.2 Quantitative Analysis

This type of analysis has dominated the financial industry in recent decades and is the foundation for modern portfolio theory, derivatives pricing, and risk management. Quantitative analysis treats asset prices as random and uses mathematical and statistical analysis to find a suitable model for describing this randomness (Nuti et al., 2011).

In the following subsections we will describe some basic calculations derived from quantitative analysis used in the following chapters.

4.4.2.1 Asset returns

The n period return of an asset is the percentage change in price and is given by the following equation:

$$R_n = \frac{P_t - P_{t-n}}{P_{t-n}} \tag{4.1}$$

where P_t is the price of the asset at time t and n the number of periods back.

4.4.2.2 Volatility

Volatility is a statistical measure of dispersion of the returns of an asset. Volatility is usually measured using the standard deviation of returns. The variance of the returns might also be used as a measure of volatility. Normally the more volatilty an asset has, the riskier it is. Volatility can be determined according to the following equation:

$$\sigma = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(R_t - \mu)^2} \tag{4.2}$$

where R_t is the return at time t and μ is the mean of the returns.

4.4.2.3 Capital Asset Pricing Model α and β

The CAPM was developed simultaneously but separately by Sharpe (1964); Lintner (1965); Mossin (1966). CAPM is one of the most widely used tools to determine the expected rate of return of an asset. The formula for calculating the expected return of an asset is as follows:

$$E(r) = R_f + \beta(E(R_m) - R_f) \tag{4.3}$$

$$\beta = \frac{Cov(R, R_m)}{Var(R_m)} \tag{4.4}$$

UNIVERSIDAD DE MÁLAGA

where R is the return of the asset, R_f is the risk-free rate, β shown in Equation 4.4 is the covariance between the returns of the asset and the market divided by the variance of the market rate of return. $E(R_m)$ is the expected return of the market.

The CAPM says the expected return of an asset or portfolio equals the risk-free rate plus a risk premium. If the expected return does not meet or improve the required return then the investment should not be considered.

Another important measure is the alpha coefficient (α) which measures an abnormal rate of return in excess of what is be predicted by CAPM. The alpha coefficient is a parameter of CAPM which can be seen as the intercept of the regression line, which is a constant in the market model linear regression.

4.4.3 Technical Analysis

Technical analysis attempts to forecast the future movement of financial instruments by analyzing data collected from trading activity, such as price and volume. Unlike fundamental analysis, whose goal is to determine a security's intrinsic value, technical analysis focuses on charts of price movement and various analytical tools to evaluate a security's strength or weakness and forecast future price changes.

"Technical analysis has been a part of financial practice for many decades, but this discipline has not received the same level of academic scrutiny and acceptance as more traditional approaches such as quantitative and fundamental analysis. However, several academic studies suggest that despite its jargon and methods, technical analysis may well be an effective means for extracting useful information from market prices" (Lo et al., 2000).

Technical analysis is based on the Dow Theory, formulated by Charles H. Dow in a series of *Wall Street Journal* editorials from 1900 to 1902. These editorials explained Dow's beliefs on how the stock market behaved. Dow believed the stock market as a whole was a reliable measure of the overall business conditions, and that by analyzing the whole market, one could accurately determine those conditions and identify the direction of major market trends and individual stocks

The following two basic assumptions of Dow Theory are the foundation of all of technical analysis:

1. Market price discounts every factor that may affect a security's price.
2. Market price movements are not purely random and move in identifiable patterns and trends that repeat over time.

The assumption that price discounts everything essentially means the market price of a security at any given point in time accurately reflects

UNIVERSIDAD DE MÁLAGA

all available information, and therefore represents the true fair value of the security. This assumption is based on the idea the market price always reflects the sum total knowledge of all market participants.

The second basic assumption underlying technical analysis, the notion that price changes are not random, and prices move in trends, both short term and long term, which can be identified. A Trend once is formed tends to persist. Trend following is one of the most typical strategies technical traders follow.

Technical analysis is used to forecast the price movement of virtually any tradeable instrument that is generally subject to forces of supply and demand, including stocks, bonds, futures and currency pairs. In fact, technical analysis can be viewed as simply the study of supply and demand forces as reflected in the market price movements of a security. It is most commonly applied to price changes, but some analysts may additionally track numbers other than just price, such as trading volume or open interest figures.

TA employs many techniques being the main one, the use of price charts where price patterns and trends can be identified graphically. TA also uses indicators and oscillators which are mathematical transformations of price and volume that help determine if there is a trend in place and its probability of continuation or change of direction (trend reversal).

Over the years, numerous technical indicators have been developed by analysts in attempts to accurately forecast future price movements. Some indicators are focused primarily on identifying the current market trend, including support and resistance areas, while others are focused on determining the strength of a trend and the likelihood of its continuation. Commonly used technical indicators include trend-lines, moving averages and momentum indicators such as the MACD (*Moving Average Convergence Divergence*) indicator.

Technical analysts apply technical indicators to charts of various timeframes Short-term traders may use charts ranging from one-minute timeframes to hourly or four-hour timeframes, while traders analyzing longer-term price movement scrutinize daily, weekly or monthly charts.

In figure 4.1 we can observe a chart of daily prices of Banco Santander's stock. The graph is subdivided in four subgraphs. The top shows price information in the form of candles. Candles represent the *Open, High, Low and Close* of every period. Open corresponds to the start of the period, while close is the price at the end of the period. High and low represent the highest and lowest price at that given period. If the close of the period is higher that the open the candle has a green color. On the other hand, if the close of the period is lower than the open, the candle has a red color. Also in the graph there are two simple moving averages of 50 and 200 periods (blue and red lines) which show the trend while filtering the noise and smoothing the time series. The second subgraph shows the volume representing the number of

UNIVERSIDAD DE MÁLAGA

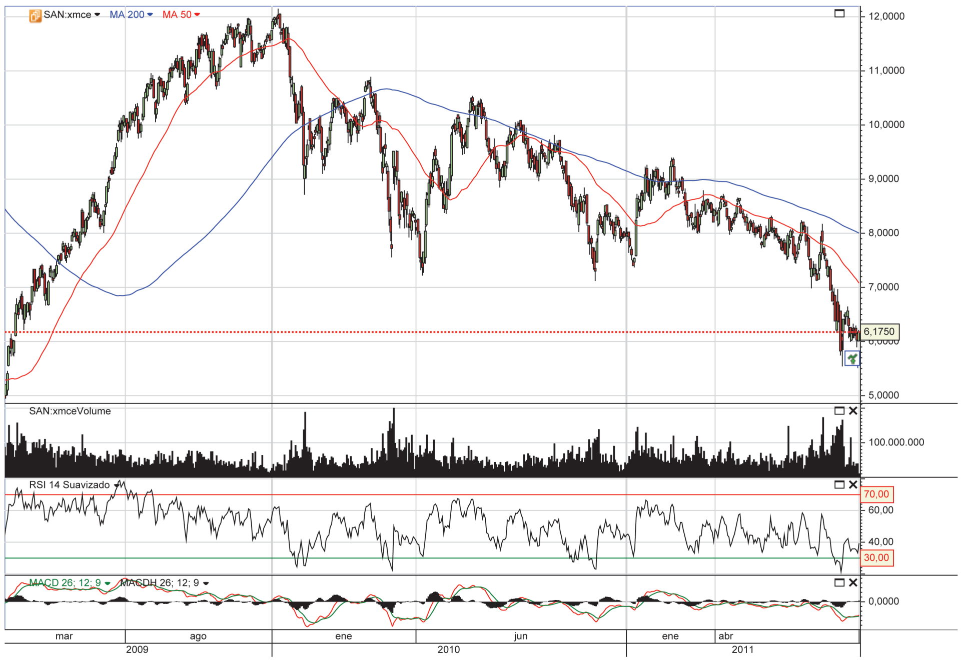


Figure 4.1: Chart of Banco Santander.

shares transacted in each period. The other two subgraphs show the RSI (*Relative Strength Index*) and MACD oscillator. The technical oscillators and indicators used as in this thesis will be discussed next.

For a detailed explanation of technical analysis and trading systems see (Murphy, 1999; Kaufman, 2005; Tsinaslanidis and Zapranis, 2016). Next we will describe the techcnical indicators we will use in the following chapters.

4.4.3.1 Moving averages

In statistics a moving average is a type on *finite impulse response filter* used to analyze a set of data points by creating a series of averages of different subsets of the full data set. There are different types of moving averages depending on how the periods are weighted. An example of a simple (unweighted) moving average of the closing prices of n days is detailed in the following equation:

$$SMA_n = \frac{P_t + P_{t-1} + P_{t-2} + \ldots + P_{t-n-1}}{n} \tag{4.5}$$

In this thesis we have also used the EMA (*Exponential Moving Average*), which applies weighting factors that decrease exponentially, but never reaching 0. This results in the most recent days having more importance than

UNIVERSIDAD DE MÁLAGA

older days. The following equation describes its calculation:

$$EMA_n = \alpha \times (P_{t-1} + (1-\alpha)P_{t-2} + (1-\alpha)^2 P_{t-3} + \ldots + (1-\alpha)^{n-1} P_{t-n}) \quad (4.6)$$

where the coefficient α represents the degree of weighting decrease. A higher α discounts older observations faster. Alternatively α may be expressed in terms of n time periods where $\alpha = 2/(n+1)$.

4.4.3.2 Moving Average Convergence Divergence

Created by Appel in the 1970s, it is used to spot changes in strength, direction, momentum and duration of a trend in a stock price. The MACD is computed by calculating the difference between two exponential moving averages of closing prices. It consists of two lines, but may be graphed also with a histogram that shows the divergence between the two EMAs, see figure 4.1. As we can see in the chart, while the price trend was up for the year 2009, we can see that the MACD lines were trending down, signalling that strength of the trend was decreasing and a peak and trend reversal where near. The first line, called the MACD line is calculated by subtracting the a long period (l) exponential moving average from a shorter (s) period exponential moving average. The signal line is the exponential moving average of g periods of the previous MACD line (Appel, 2005). The equations are:

$$MACD\ line = EMA_l(Close) - EMA_s(Close) \quad (4.7)$$

$$Signal\ line = EMA_g(MACD\ line) \quad (4.8)$$

4.4.3.3 Relative Strength Index

Developed by Wilder (1978), the RSI (*Relative Strength Index*) is intended to capture the current and historical strength or weakness of a stock based on the closing prices of a recent period of length n. The RSI is a momentum oscillator, measuring the velocity and magnitude of directional price movements. Momentum is defined as the rate of the rise or fall in price. The RSI computes momentum as the ratio of higher closes to lower closes: stocks which have had more or stronger positive changes have a higher RSI than stocks which have had more or stronger negative changes. The RSI is most typically used on a 14 day timeframe, measured on a scale from 0 to 100, with high and low levels marked at 70 and 30, respectively. In figure 4.1 we can see how the RSI in the second half of 2009 was trending down while

UNIVERSIDAD DE MÁLAGA

price was trending up, this is a signal of possible trend reversal as the trend was loosing momentum. The calculation is the following:

$$\text{if } Close_t > Close_{t-1} \Rightarrow UP_t = Close_t - Close_{t-1} \text{ and } DOWN_t = 0 \quad (4.9)$$

$$\text{if } Close_t < Close_{t-1} \Rightarrow DOWN_t = Close_{t-1} - Close_t \text{ and } UP_t = 0 \quad (4.10)$$

$$RSI = \frac{EMA_n(UP)}{EMA_n(UP) + EMA_n(DOWN)} \times 100 \quad (4.11)$$

4.4.3.4 Highest High and Lowest Low

HH (*Highest High*) and LL (*Lowest Low*) represent the maximum and minimum prices n periods ago and are calculated according to the following equations:

$$HH_n = max(H_t, \ldots, H_{t-n}) \quad (4.12)$$

$$LL_n = min(L_t, \ldots, L_{t-n}) \quad (4.13)$$

where H_t is the $High$ and L_t is the Low at time t in a OHLCV (*Open, High, Low, Close, Volume*) price time series.

4.4.3.5 Stochastic Oscillator

Developed by Lane in the 1950s, the SO (*Stochastic Oscillator*) measures the momentum of the trend. It consists of two lines $\%K$ and $\%D$ and refers to the location of the current price in relation to its price range during the past n periods. The calculation finds the range between the high and low price during the specified period of time. The price is then expressed as a percentage of this range with 0% indicating the bottom of the range and 100% indicating the upper limits of the range over the time period covered. The idea behind the SO is that prices tend to close near the extremes of the recent range before reversing. The following equations detail its calculation:

$$\%K_n = (P_t - LL_n)/(HH_n - LL_n) \quad (4.14)$$

$$\%D_n = EMA_j(\%K) \quad (4.15)$$

where P_t is the price at time t, LL_n is the *Lowest Low*, HH is the *Highest High* and n the number of periods back and EMA_j is the exponential moving average using j periods back (Lane, 1984).

UNIVERSIDAD DE MÁLAGA

4.4.3.6 Bollinger Bands ®

Invented by Bollinger in the 1980s and trademarked by him in 2011, BB (*Bollinger Bands* ®) is a volatility indicator consisting of:

- an n-period SMA (*Simple Moving Average*)
- an upper band at k times an n-period standard deviation above the simple moving average $(SMA + k\sigma)$
- a lower band at k times an n-period standard deviation below the simple moving average $(SMA - k\sigma)$

The goal of BB is to provide a relative reference to high and low values. Prices are high at the upper band and low at the lower band. Some traders buy when the price touches the lower band and exit when the price touches the upper band. Others buy when the price breaks above the upper band and sell when the price falls below the lower band (Bollinger, 2001).

4.4.3.7 Internal Bar Strength

IBS (*Internal Bar Strength*) is based on the position of the close in relation to the day's range: it takes a value of 0 if the closing price is the lowest price of the day, and 1 if the closing price is the highest price of the day. The following equation details its calculation:

$$IBS = \frac{Close - Low}{High - Low} \tag{4.16}$$

4.5 Trading strategy performance metrics

There are a wide variety of metrics for assessing the performance of trading strategies, being the most widely used the Sharpe ratio, the Sortino ratio, the Sterling ratio and total return (Iba and Aranha, 2012). Next we will discuss the most important ones mentioned in this thesis.

4.5.1 Total return

Total return measures the actual rate of return in percentage of an asset or portfolio over a given period. It is the simplest performance calculation and does not contain any risk component. Total return is calculated as shown earlier in Equation 4.1.

UNIVERSIDAD DE MÁLAGA

4.5.2 Sharpe ratio

The *Sharpe ratio* (Sharpe, 1966, 1994) provides a risk-adjusted measure of the performance of an asset or portfolio. The following equation details its calculation.

$$\textit{Sharpe Ratio} = \frac{\mu - R_f}{\sigma}\sqrt{n} \tag{4.17}$$

where μ is the mean of the asset or portfolio returns, R_f is the risk-free rate, σ is the volatility or standard deviation of returns and n the number of observations.

4.5.3 Sortino ratio

The *Sortino ratio* (Sortino and Price, 1994) is a modification of the Sharpe Ratio that does not penalize positive outlier returns. The Sortino Ratio only penalizes negative returns and is calculated as follows:

$$\textit{Sortino Ratio} = \frac{\mu - R_f}{\sigma_{R<0}}\sqrt{n} \tag{4.18}$$

where $\sigma_{R<0}$ is the downside risk measured by the standard deviation of negative returns.

4.5.4 Sterling ratio

The *Sterling ratio* is another risk-adjusted performance measure similar to the Sharpe and Sortio ratios and is calculated as:

$$\textit{Sterling Ratio} = \frac{R - R_f}{\text{Maximum Drawdown}} \tag{4.19}$$

where R is the total return (Equation 4.1), R_f the risk-free rate and *Maximum Drawdown* is the maximum decline in portfolio value from peak to nadir measured as percentage return.

4.5.5 Maximum drawdown

The *Maximum Drawdown* is a metric of risk that measures the maximum loss from a peak to a trough of a portfolio or asset before a new peak is attained. Assuming R_t is the total return at time t, $(t \geq 0)$ the drawdown at time n, is defined as:

$$\textit{Drawdown} = \max\left\{0, \max_{t\in(0,n)} R_t - R_n\right\} \tag{4.20}$$

UNIVERSIDAD DE MÁLAGA

the *Maximum Drawdown* up to time n is the maximum of the drawdowns over the history of returns,

$$\textit{Maximum Drawdown} = \max_{\tau \in (0,n)} \left[\max_{t \in (0,\tau)} R_t - R_\tau \right] \tag{4.21}$$

4.6 Conclusions

In this chapter we have introduced algorithmic trading and explained the different components that form and algorithmic trading system, together with some basic notions on the three most widely used financial methodologies used for pre-trade analysis. In the next chapters we employ the technical and quantitative analysis calculations described as the foundation for the strategies we will optimize. We have also explained the most common metrics for measuring the performance of trading strategies that will be used in the following chapters.

UNIVERSIDAD DE MÁLAGA

Part II

Problems addressed

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA

Chapter 5

Robust optimization of technical trading strategies

All of technology, really, is about maximizing free options.

Nassim Nicholas Taleb

5.1 Introduction

TTSs (*Technical Trading Strategies*) are widely used by traders and over the past two decades attracted a lot of attention from researchers (Aguilar-Rivera et al., 2015; Hu et al., 2015). TTSs are defined as buy and sell rules derived from TA (*Technical Analysis*) which can be very useful (Brock et al., 1992; Gencay, 1998; Lo et al., 2000) but often involve different parameters, which need to be determined, that greatly influence the quality of the buy and sell signals obtained.

TTS optimization is very similar to a machine learning problem, where we are trying to learn the best parameter set from a historic dataset. In order to determine how the parameters found during optimization would work in the real-world when deployed, we need to split the datasets into a training and evaluation datasets as its typically done in machine learning tasks.

One of the most typical problems practitioners face when optimizing the parameters of a TTS is over-fitting. Very good solutions are found during the optimization phase for the training data, but most of these solutions perform poorly when tested out-of-sample. This problem arises as a consequence of selecting the best performing parameters without regarding their robustness and stability, thus being very fragile to small disturbances in the data.

Market conditions are dynamic and continuously evolving, and when the ex-optimal parameter settings are applied to the out-of-sample period, any small disturbance in the data results in great performance degradation. In

UNIVERSIDAD DE MÁLAGA

this chapter we try to address the following questions:

- Is it possible to find a robust parameter setting that works well for both the in-sample and out-of-sample periods?
- How may we accomplish this?

The main contribution of this chapter is a robust multi-objective approach using a GA that involves optimizing the same parameter settings over multiple market indices, therefore increasing the different environmental conditions where the proposed solution has to operate reliably. Our proposed method of optimization is capable of producing a unique set of robust parameters that not only works well over the different markets for the in-sample data, but also has a similar or better level of performance when tested out-of-sample.

Markets all have different intrinsic peculiarities, but also share many common characteristics. Increasing the dataset to include several indices, exposes the profitable recurring patterns present over the various markets and helps guide the GA towards them. On the other hand, optimizing over only one financial instrument inexorably directs the algorithm to the singular traits of that sole market, while reducing the exposure to different variable market conditions thus over-fitting most solutions.

5.2 Literature review

GAs were fist used in financial applications by Bauer and Liepins (1988); Bauer (1994) and have become a very popular optimization method due to their ability to solve a great variety of financial engineering problems (Chen, 2002; Chen et al., 2007; Brabazon et al., 2012; Iba and Aranha, 2012; Aguilar-Rivera et al., 2015).

As the main concern of this chapter is robust optimization of TTS, the scope of this section is limited to the most relevant papers that, in our opinion, deal specifically with robustness in TTS using diverse methods.

One of the first works is Pictet et al. (1995) where they optimize a TTS for the FX (*Foreign Exchange*) markets. Robustness is incorporated into the fitness function of a GA by using a *K-Means* clustering algorithm and a specifically designed fitness sharing scheme between individuals. This optimization methodology is able to produce robust parameters that correspond to broad regions of the search space where fitness is higher on average.

Ni et al. (2008) optimize TTSs on 32 stocks and 4 indices. They deal with the issue of robustness in the fitness function taking into consideration its resilience to parameter disturbance. This is what is known in robust optimization as a *parameter sensitivity analysis*. This method is able to

UNIVERSIDAD DE MÁLAGA

evolve robust parameters and solves over-fitting, displaying stable performance when tested out-of-sample.

Matsui and Sato (2010) propose a similar robust methodology where the parameters are evaluated along a neighborhood of values. Sharp peaks of high fitness are not indicative of good solutions, but rather indicate noise inherent in financial time series. Evaluating the neighboring points in the fitness landscape, as well as itself, ensures a reduction of the influence of singular points during optimization and improves performance during out-of-sample testing.

Li and Taiwo (2006) employ a different approach borrowed from supervised learning for evolving their trading models with multi-objective genetic programming. The authors propose an optimization methodology for improving generalization whereby the dataset is split in three sections: optimization, validation and evaluation. Both optimization and validation sections of the dataset are made visible to the algorithm and fitness values are calculated on each part. The fitness function receives only the worst value of the two sections, thus limiting over-fitting. When the optimization ends, the solutions are finally tested out-of-sample in the evaluation section of the dataset.

Pinto et al. (2015) use a dual-objective genetic algorithm to maximize the total return on investment (ROI) while minimizing the standard deviation of returns. In their study they introduce the VIX volatility index and other technical indicators to boost performance of trading strategies for stock market indices to optimize the parameters of the strategies. This approach is able to avoid serious market declines, while producing a Pareto front of non-dominated solutions that can cater from the most conservative types of investors looking for strategies with minimal risk to the most aggressive ones, who prefer higher returns at a higher risk. Our proposed method is similar but has one key advantage as our system optimizes the parameters of a technical trading strategy in multiple of stock indices, instead of a single market index, thus exposing the evolutionary process to more data, and different market conditions, as some indices can be in a bullish trend while other are bearish or range bound (sideways), producing solutions that are robust and able to cope with extreme market conditions.

The above mentioned research deals with robustness considering how the variability in the parameter values, i.e. the *design variables*, affects the performance of the system. Our proposed method, on the other hand, is based o a non-deterministic approach affecting the *environmental variables* of the problem as the search is conducted over many markets. This is similar to what Pardo (2008) hints when he offers the following tip: “The more markets that a model can trade, the more useful it is”.

We use a GA to perform a multi-objective optimization of the parameters of a trading strategy over 12 of the most important European and American

UNIVERSIDAD DE MÁLAGA

stock market indices. Our results establish evidence about the benefits of using this method and how it leads to more robust solutions that perform well out-of-sample.

5.3 The robust TTS problem

This section elaborates on the optimization problem addressed in this chapter. We first provide a brief description of financial time series, followed by the TTS used, and how it is designed. We then formulate the task of finding the best parametrization of our TTS as an optimization problem, and finally, we describe how we have endowed this problem with robustness, so as to devise a TTS being able to profit from any market scenario.

5.3.1 Investment vehicle

In the study conducted in this chapter we chose future contracts on stock market indices as our financial vehicle due to their implicit diversification, high liquidity and reduced commissions. Indexes are composed of a basket of the most liquid and biggest listed companies in terms of capitalization that are traded in a given market. Trading futures has the key advantage of allowing taking both long and short positions easily, while being more cost efficient than a CFD (*Contract For Differences*) or an ETF (*Exchange Traded Fund*) that track an index.

5.3.2 Technical trading strategies

We have based our TTS on previous works by (Subramanian et al., 2006) and (Briza and Naval, 2011) where a series of technical indicators are used in a weighted manner. We have chosen five popular technical indicators, EMA, MACD, RSI, SO and the HH and LL price of n periods ago.

Each technical indicator forms a component rule that yields a trading signal $s_i \in \{-1, 0, 1\}$ which can have three values: *sell*, *do nothing* and *buy* respectively. A position is maintained until there is a change in the signal value. For example if at time t we have $s_i = 1$ and at time $t + 1, s_i = 1$ then we would maintain the same long position. If at time $t + 1$, the signal changes to $s_i = 0$, we would close the long position and remain out of the market, while if the signal changed to $s_i = -1$, it would mean closing the long position and initiating a new short position.

We will now explain each of these component strategies and how we arrive to a combined decision.

UNIVERSIDAD DE MÁLAGA

5.3.2.1 Exponential Moving Average

In this component rule, we use the first-order difference or approximate derivative of the EMA vector, represented as ΔEMA, and calculated as the difference between its adjacent elements: $\Delta EMA_1 = 0, \Delta EMA_t = ema(n)_t - ema(n)_{t-1}$. We use it to measure the speed of the EMA and detect when it slows down. We generate the signal vector for this component rule according to:

$$s_1 = \begin{cases} 1 & \text{if} \quad \Delta EMA(n) > k \\ -1 & \text{if} \quad \Delta EMA(n) < -k \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

where k is used as a threshold value. This component rule has two parameters, k and n (the EMA's number of periods), that need to be optimized.

5.3.2.2 Moving Average Convergence Divergence

The MACD is used to spot changes in strength, direction, momentum and duration of a trend. This component rule uses three parameters, the periods for the three different EMAs, that need optimization and is generated as follows:

$$s_2 = \begin{cases} 1 & \text{if} \quad MACD > Signal \\ -1 & \text{if} \quad MACD < Signal \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

.

5.3.2.3 Relative Strength Index

We have used a detrendend version (Lorenzo, 2012) of the price series to calculate the RSI, where a exponential moving average of the closing price is subtracted from the original price series. The original RSI calculation is then applied to the detrended price series.

This component rule has three parameters, the number of periods of the EMA for calculating the detrended series, as well as the number of periods for the RSI calculation and a threshold value k so we can customize the default 70/30 high and low levels. This trading signal is generated as:

$$s_3 = \begin{cases} 1 & \text{if} \quad RSI < k \\ -1 & \text{if} \quad RSI > 100 - k \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

UNIVERSIDAD DE MÁLAGA

5.3.2.4 Stochastic Oscillator

This oscillator measures the momentum of the trend. This rule has three parameters, the number of periods for the $\%K$ and $\%D$ lines and the k threshold, and it is calculated as follows:

$$s_4 = \begin{cases} 1 & \text{if} \quad \%K < \%D \\ & \text{and} \quad \%K < k \\ & \text{and} \quad \Delta\%D > 0 \\ \\ -1 & \text{if} \quad \%K > \%D \\ & \text{and} \quad \%K > 100 - k \\ & \text{and} \quad \Delta\%D < 0 \\ \\ 0 & \text{otherwise} \end{cases} \tag{5.4}$$

5.3.2.5 Highest High and Lowest Low

This rule is very simple. If we have increasing high and low values we have a buy signal. When both are decreasing, we signal a sell. This component rule is generated according to:

$$s_5 = \begin{cases} 1 & \text{if} \quad (h \geq HH) \text{ and } (l > LL) \\ -1 & \text{if} \quad (h < HH) \text{ and } (l \leq LL) \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

where h and l are the current interval high and low prices respectively. This component rule uses two parameters, the number of periods for the HH and LL calculations.

5.3.2.6 Combining the component rules

The five component signals s_i are combined in a weighted manner and then compared to a threshold k to form the final decision signal:

$$S = \begin{cases} 1 & \text{if} \quad \sum \omega_i \cdot s_i > k \\ -1 & \text{if} \quad \sum \omega_i \cdot s_i < -k \\ 0 & \text{otherwise} \end{cases} \tag{5.6}$$

where $0 \leq \omega_i \leq 1$ and $0 \leq k \leq i$.

Having a weighted decision scheme permits giving more importance to the best component signals. The weights ω_i and the threshold k together with the rest of parameters from the previous component rules, 19 in total, will be optimized.

UNIVERSIDAD DE MÁLAGA

5.3.2.7 Strategy simulation

To calculate the return we presume that we enter the market at the closing price when signal S is given. Given a vector $\vec{dC} = \Delta Close$ calculated as the first order difference of the closing price vector; $dC_1 = 0$, $dC_t = Close_t - Close_{t-1}$ and vector $\vec{dS} = \Delta S$ being the first order difference of the signal vector S; $dS_1 = 0$, $dS_t = S_t - S_{t-1}$. We calculate the absolute returns of our strategy as vector $\vec{r}$ at time t as:

$$r_t = S_t \cdot (dC_t \cdot \eta) - |dS_t \cdot \gamma/2| \tag{5.7}$$

where $r_1 = 0$ and η is the size of each futures contract. For example one point of a IBEX35 future contract is equivalent to 10 €; γ is the round-trip commission and slippage associated with the asset. We trade only one contract and do not reinvest profits. We assume no leverage and a starting capital equal to the contract's full nominal value at the starting period.

5.3.3 Objective functions to be optimized

We have previously explained the decision variables of our optimization problem. Now we need to determine the evaluation criteria.

The selection of the fitness function is crucial, as it is the means of comparison between different parameters, and the deciding factor for selecting the best combination. If we selected the total profits earned by the strategy as our criteria, it would result in solutions that ignore risk. If two strategies are compared, one with a return of 25% and another with 24%, the algorithm would choose the one with the highest value, even though the chosen solution could have considerably more volatility than the other one, and be in practice a worse solution.

This is why we have selected a risk adjusted measure as the $\mathcal{SR}$ (*Sortino Ratio*). The $\mathcal{SR}$ is a variation of the Sharpe ratio that has the advantage of not penalizing positive volatility. We scale the $\mathcal{SR}$ in all our calculations multiplying it by the $\sqrt{n}$ where n is the number of observations in the price series. This is done in order to be able to compare ratios of series that may have different lengths (Lo, 2003).

Another important factor affecting the usefulness of a TTS is the $\mathcal{MDD}$ (*Maximum Drawdown*) or loss that can be incurred. Any sane investor would avoid a TTS with a large maximum drawdown. With these two objective functions, i.e., maximization of $\mathcal{SR}$ and minimization of $\mathcal{MDD}$, several optimization problems have been formulated always targeting the definition of a robust TTS, though some of them are used just as a comparison basis.

UNIVERSIDAD DE MÁLAGA

5.3.3.1 Single-market optimization of $\mathcal{SR}$

This first optimization problem is stated to settle a comparison basis. It is solely based on the optimization of $\mathcal{SR}$ separately on each of the n different markets considered in this chapter, i.e., there are n different instances of this optimization problem. More formally, let $\mathcal{SR}$ be the sortino ratio of a TTS calculated in a specific market $\mathcal{M}_i$ with parameters $\vec{x}$,

$$\max \mathcal{F}(\mathcal{M}_i, \vec{x}) = \mathcal{SR}(\mathcal{M}_i, \vec{x}), \quad i = 1, 2, ...n \ . \tag{5.8}$$

where n is the number of different markets.

A similar problem could have been defined by using $\mathcal{MDD}$ as a single function, but we have obviated this option, as it does not make any sense to consider a trading strategy without any profit indicator. Optimizing just $\mathcal{MDD}$ would lead to an optimal strategy in which no trading is performed, and therefore no loss is obtained.

5.3.3.2 Robust multi-objective optimization of $\mathcal{SR}$

In the previous section, a rather standard optimization problem of the TTS's parameters over a single market has been proposed. In this section, we take the first step on the quest for a robust TTS parameterization. The goal of robust optimization is finding solutions which are immune to production tolerances, parameter drifts and model sensitivities (Beyer and Sendhoff, 2007). Our approach aims at obtaining a TTS with a robust $\mathcal{SR}$ by evaluating such a trading strategy over several markets, i.e. our approach looks for set of TTS parameters that performs well over many different market scenarios. As stated in Section 5.1, markets have their own distinctive attributes, but also share common characteristics. The goal here is that a TTS will be able to learn these similarities and will not degrade its performance once it is deployed.

The evaluation of a TTS over different markets leads to a set of $\mathcal{SR}$ values, one value for each market. The usual approach in the specialized literature (as we have seen in Chapter 3) is to aggregate these values somehow (Deb and Gupta, 2006), being the mean the most widely used central tendency measure. However, we are not only interested in obtaining a TTS which has the best average performance over several markets, but also the dispersion of these values should be kept as small as possible, thus indicating that it behaves in a similar way, i.e., it is not very sensible to market conditions. This is our motivation for using the standard deviation of the different $\mathcal{SR}$ values obtained. This mechanism has been also used in the literature (Beyer and Sendhoff, 2007). As a consequence, the problem is now formulated as a bi-objective optimization problem:

UNIVERSIDAD DE MÁLAGA

$$\begin{aligned}
\max \mathcal{F}_1(\mathcal{M}_i, \vec{x}) &= \sum_{i=1}^{n} \frac{\mathcal{SR}(\mathcal{M}_i, \vec{x})}{n} \\
\min \mathcal{F}_2(\mathcal{M}_i, \vec{x}) &= \sqrt{\sum_{i=1}^{n} \frac{(\mathcal{SR}(\mathcal{M}_i, \vec{x}) - \mathcal{F}_1(\mathcal{M}_i, \vec{x}))^2}{n}}
\end{aligned}$$

where $\vec{x}$ are the TTS parameters, n is the number of different markets. Recall that here we are just defining one single problem instance (that considers all the available markets), whereas in the previous formulation we have one for each market.

5.3.3.3 Single-market optimization of both $\mathcal{SR}$ and $\mathcal{MDD}$

In this formulation of the problem, the maximum drawdown, $\mathcal{MDD}$, comes into play. As stated above, it is a TTS performance measure of great relevance for any investor. We proceed here as in Section 5.3.3.1 by defining an optimization problem that operates on a single market, which will be extended afterwards in order to look for a more robust TTS. If $\vec{x}$ are the parameters of the TTS, then the optimization is formally defined as:

$$\begin{aligned}
\max \mathcal{F}_1(\mathcal{M}_i, \vec{x}) &= \mathcal{SR}(\mathcal{M}_i, \vec{x}) \\
\min \mathcal{F}_2(\mathcal{M}_i, \vec{x}) &= \mathcal{MDD}(\mathcal{M}_i, \vec{x})
\end{aligned}$$

where $i = 1, 2, \ldots, n$ points out the particular market where $\mathcal{SR}$ and $\mathcal{MDD}$ are optimized. Therefore, n problem instances have to be solved.

5.3.3.4 Robust multi-objective optimization of both $\mathcal{SR}$ and $\mathcal{MDD}$

The last optimization problem defined in order to devise a robust trading strategy, considers both $\mathcal{SR}$ and $\mathcal{MDD}$, but averaged over n different markets. As in Section 5.3.3.2, the goal is to search for a robust TTS parameterization which is known to perform well over different market conditions. Our hypothesis is that the resulting TTS will not only avoid over-fitting, but also degrading its performance when facing new unseen market scenarios. Having $\mathcal{SR}$ and $\mathcal{MDD}$ evaluated over n markets leads to a four-objective problem that considers the mean and standard deviation, as measures of central tendency and dispersion, of these two trading performance indicators. Formally:

UNIVERSIDAD DE MÁLAGA

$$\begin{aligned}
\max \mathcal{F}_1(\mathcal{M}_i, \vec{x}) &= \sum_{i=1}^{n} \frac{\mathcal{SR}(\mathcal{M}_i, \vec{x})}{n} \\
\min \mathcal{F}_2(\mathcal{M}_i, \vec{x}) &= \sqrt{\sum_{i=1}^{n} \frac{(\mathcal{SR}(\mathcal{M}_i, \vec{x}) - \mathcal{F}_1(\mathcal{M}_i, \vec{x}))^2}{n}} \\
\min \mathcal{F}_3(\mathcal{M}_i, \vec{x}) &= \sum_{i=1}^{n} \frac{\mathcal{MDD}(\mathcal{M}_i, \vec{x})}{n} \\
\min \mathcal{F}_4(\mathcal{M}_i, \vec{x}) &= \sqrt{\sum_{i=1}^{n} \frac{(\mathcal{MDD}(\mathcal{M}_i, \vec{x}) - \mathcal{F}_3(\mathcal{M}_i, \vec{x}))^2}{n}}
\end{aligned}$$

where x are the TTS parameters, n is the number of different markets. One single instance of this problem has to be addressed.

5.4 Algorithmic approach

The search for a trading strategy's optimum parameters is an optimization problem that is well suited for employing meta-heuristic methods that do not get trapped in local optima, as the search space can be highly non-linear and discontinuous. GAs are non-deterministic algorithms that have great application for solving complex search and optimization problems and machine learning (Goldberg, 1989).

We have choosen the MATLAB® Global Optimization Toolbox (R2012b) implementation under the function `gamultiobj`, which uses a controlled elitist variant of NSGA-II (Deb et al., 2002). NSGA-II is a multi-objective GA that is characterized by two features: the use of a *Pareto ranking* mechanism to classify solutions, and a density estimator known as *crowding distance*. The ranking of solutions classifies a population in ranks (1, 2, ...) in such a way that the non-dominated solutions are assigned a rank equal to 1; then, they are removed and the procedure is successively applied yielding to solutions with ranks 2, 3, and so on. By selecting the solutions with best ranking, NSGA-II tries to converge towards the true Pareto front. However, when choosing the best ranked solutions it is possible that only a subset of solutions of a given rank be needed. In this case, it is necessary to carefully select the most promising solutions in order to promote diversity, and the approach taken in NSGA-II is to define a *density estimator*. The idea of a density estimator is to assign to a set of non-dominated solutions a value indicating in some way the degree of proximity (or density) of nearby solutions in the set. This way, solutions in sparse regions are preferred compared to those in most crowded regions. As indicated before, the density estimator in NSGA-II is called crowding distance. NSGA-II has become

UNIVERSIDAD DE MÁLAGA

de *de facto* multi-objective optimization meta-heuristic. An outline of the method is displayed in Algorithm 2.

Algorithm 2 Pseudocode of the NSGA-II algorithm.

```
1: $P(0)$ GenerateInitialPopulation()
2: $t \leftarrow 0$
3: Evaluate($P(0)$)
4: while not StoppingCriterion( ) do
5:     $P'(t) \leftarrow$ BinaryTournament($P(t)$)
6:     $P''(t) \leftarrow$ Crossover & Mutation($P'(t)$)
7:     Evaluate($P''(t)$)
8:     $P(t+1) \leftarrow$ Ranking & NonDomintatedSorting ($P''(t)$)
9:     $t \leftarrow t+1$
10: end while
```

We have used the MATLAB® implementation of NSGA-II for two main reasons being the first one strategic. On the one hand, we do not want to focus the attention of the readers on any new algorithmic approach specially tailored to the problem addressed, but to the method proposed for designing robust TTS. On the other hand, convenience as MATLAB® has allowed us to use both its financial toolbox (to manage market data and enhanced visualization tools) together with the parallelization capabilities on multiple cores which results in a reduction of computational time. MATLAB® is also quite common across the research community hence results may be easily reproduced by others.

5.4.1 Problem encoding

We encode the 19 decision variables in a vector of real numbers. Our problem has some parameters that are integers such as the EMA periods, where rounding was employed. The parameters that serve as thresholds and weights are real numbers.

The encoding for the parameters of our TTS mentioned in Section 5.3 is very straight forward. For example the first two parameters from the EMA rule s_1 correspond to x_1 and x_2. The next 3 parameters from the MACD rule s_2 correspond to x_3, x_4 and x_5. The parameters for the RSI rule s_3 correspond to x_6, x_7 and x_8, and so on.

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	$\ldots$	x_{19}

Figure 5.1: Chromosome of the decision variables

UNIVERSIDAD DE MÁLAGA

5.4.2 Genetic operators

The operators we use are directly available in the toolbox solver. After some preliminary experimentation, we selected the ones that suited our problem best. As each of the parameters in our problem is encoded in a specific gene position of the chromosome, our choice of operators is limited to those that can respect its position and constraints.

We chose *tournament selection* where two individuals are chosen at random and the best individual is kept as the first parent (the process is repeated to select the second parent). For reproduction we use *uniform crossover*. This operator creates a random binary vector that is used as a selection mask for choosing from which parent each gene is coming. For example, to form the new child the genes where the vector is a 1 would come from the first parent, and the genes where the vector is a 0 from the second parent. Figure 5.2 shows its operation.

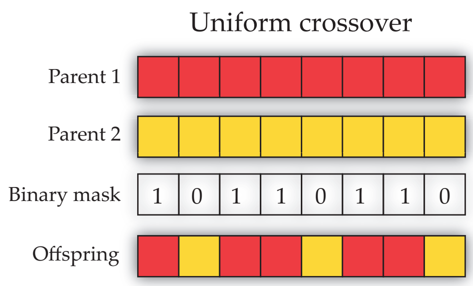


Figure 5.2: Uniform crossover operator

The *Adapt feasible mutation* operator is used. This mutation operator takes into account linear and bound constraints while generating new mutated individuals. First it generates random mutation direction vectors and random initial step sizes. A mutated individual is then generated and moved along a randomly chosen mutation direction vector with distance equal to the initial step size. The generated mutated individual is compared to the linear and bound constraints. In the event the generated mutated individual is located in an infeasible region, the operator adjusts the step size to a smaller value and generates another mutated individual along the chosen mutation direction vector. The process iterates until the produced individual is within the feasible region. The number of available valid mutation directions increases as the step size decreases.

5.5 Methodology

In this section we explain our robust multi-market optimization testing approach. To verify its usefulness, we compare the results with those obtained

UNIVERSIDAD DE MÁLAGA

using single-market optimization.

5.5.1 Dataset used

Our dataset consisted of 12 of the most important stock market indices from the year 2000 until September 2012, using OHLCV sampled at 30 minute intervals. The dataset is split in two sections. One part of the data (80%) is used to search for the best solution, the in-sample data. The obtained solution is then evaluated against new data that wasn't previously available during the optimization or search phase, the out-of-sample data (20%). The training data period is 10.2 years, while evaluation was 2.5 years. Three of the indices (AEX25, FTSE100 and SMI30) had smaller datasets of 9 years starting in 2003 resulting in 7.2 years for training and 1.8 years for evaluation.

The selection of the indices we used was conditioned by various factors. Intra-day market data is not easily available for free, we used VisualChart® which offers free intra-day access for only some indices after markets close. Our selection was limited to those indices freely available, and whose dataset length was similar. Some Asian and South American markets had only 4 to 5 years of data available, and the data had some errors, so we were forced to exclude them.

Table 5.1 lists the indices used, contract size (point equivalence in money terms), round-trip transaction costs (commissions plus slippage), and period covered by the datasets (MM/YY). For the NASDAQ 100, NASDAQ Composite and SP500 indices, we have used the *e-mini* contract.

Index	Country	Size	Cost	Years Covered
AEX25	NL	200	5€	07/2003 to 09/2012
CAC40	FR	10	5€	01/2000 to 09/2012
DAX	DE	25	5€	01/2000 to 07/2012
DJIA	US	10	2$	05/2000 to 09/2012
FTSE100	UK	10	4£	10/2003 to 09/2012
IBEX35	ES	10	5€	01/2000 to 09/2012
MIBTEL	IT	5	5€	05/2000 to 09/2012
NAS100	US	20	2$	01/2000 to 09/2012
NASCOMP	US	20	2$	08/2000 to 09/2012
SMI30	CH	10	CHF 5	12/2003 to 09/2012
SP500	US	50	2$	01/2000 to 09/2012
STOXX50	EU	10	5€	01/2000 to 09/2012

Table 5.1: Stock market indices covered in this chapter

UNIVERSIDAD DE MÁLAGA

5.5.2 GA Settings

The following settings have been applied in all our experiments. Scattered crossover was performed on 80% of the population, Adapt Feasible mutation is applied on the remaining individuals not selected for crossover apart from the elite children. An elite count of 5% of the population is used to preserve the best individuals. A population size of 100 individuals was used. The initial population was randomly created on every execution. Lower and upper bounds were established on the parameter values in order to limit the search space and to produce valid solutions. The algorithm was executed a maximum of 50 generations. Table 5.2 shows a summary of the GA settings used in all the experiments of the chapter.

Parameter	Setting
Population size	100
Selection Operator	Binary tournament
Crossover Operator	Scatter crossover $P_c = 0.80$
Mutation Operator	Adapt feasible mut. $P_m = 0.15$
Elite Count	0.05
Max generations	50

Table 5.2: GA parameter settings

5.5.3 Computational test environment

Our test machine consisted of a dual Intel Xeon CPU E5-2687W @ 3.10GHz running Ubuntu 12.04 Linux with 64Gb of RAM. These processors have 8 cores each with hyper-threading giving a total of 32 cores. To fully utilize this processing power, we used MATLAB R2012b Distributed Computing Toolbox. This was done to circumvent the limitation of a maximum of 12 workers in the Parallel Computing Toolbox.

Single execution times using 32 MATLAB workers averaged around 4 minutes for the optimizations in a single market, and 35 minutes for multi-market optimizations.

5.6 Experimental results

In this section, we describe the experiments with our robust multi-market method and compare the results with those obtained in single-market optimization.

UNIVERSIDAD DE MÁLAGA

5.6.1 Experiment 1: Single-Market Optimization of $\mathcal{SR}$

This first experiment consisted in optimizing each market independently to maximize the sortino ratio, $\mathcal{F} = \mathcal{SR}$.

Our motivation behind this experiment is showing how solutions generated with this method suffer from over-fitting. This experiment also serves as the basis of comparison with Experiments 2 and 3. In Experiment 2, we introduce our proposed robust methodology and compare the results with those obtained here. In Experiment 3, we incorporate the $\mathcal{MDD}$ into a single market multi-objective problem and examine the results to verify whether this additional objective offers any value.

We ran the GA 30 times for every market. The results obtained in this experiment are summarized in Figure 5.3, showing a box plot comparing training and evaluation results. We have performed a Kolmogorov-Smirnov (KS) test, which is a non-parametric test on the results obtained during training and evaluation, rejecting the null hypothesis that the results follow a normal distribution at the 95% confidence level. This is our motivation behind displaying results in box plots, as comparing the means of non-normal distributions does not give enough information to assert that the performance from one group is better than the other due to the skewness and kurtosis of their distributions.

A robust solution could be characterized as a solution whose performance is similar during both training and evaluation phases. Ideally, the solutions produced after the optimization phase should be robust, and have a similar performance when tested out-of-sample. Otherwise, if there is a big difference between the training and testing performance, the solutions can not be trusted as they do not have a predictable behaviour. Previous studies by (Chiam et al., 2009) mention the importance of having a positive correlation between training and evaluation performance. We measured the correlation between $\mathcal{SR}$s obtained during training and evaluation. As the results of all the experiments do not follow a normal distribution, Spearman rank correlation analysis was performed. We provide Table 5.3 where Spearman correlation coefficient ρ and $Pvalues$ are given; ρ measures the level of correlation between the variables, and it can have values in the range (-1,1), while $Pvalues$ measures the confidence level for testing the hypothesis of no correlation against the alternative that there is a non-zero correlation.

In (Mehta and Bhattacharyya, 2004), *shrinkage* is introduced, as "*a more appropriate measure of over-fit, measured as the percentage change in performance from training data to the test data, thereby revealing the degree of over-fit to noise in the training*".

We calculated shrinkage and provide a summary in Table 5.4 with mean and standard deviation for all the different experiments.

The $\mathcal{SR}$s obtained with this procedure were very high for the in-sample

UNIVERSIDAD DE MÁLAGA

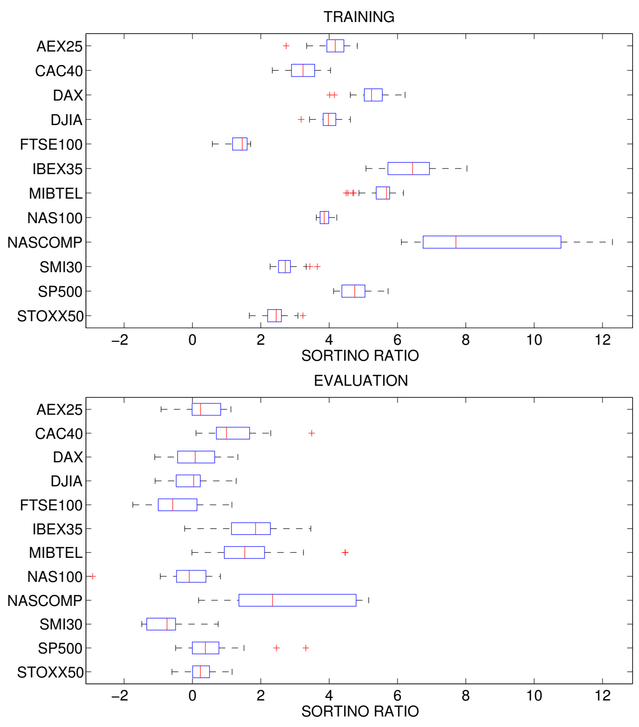


Figure 5.3: Experiment 1. Single-Market Optimization of $\mathcal{SR}$. Results for training and evaluation.

data for most indices, but not so when tested during evaluation. This is a clear sign of over-fitting, and is visible in Figure 5.3, as most indices scored high during training, but poorly during evaluation. This is also evident in Table 5.3 by the negative correlations on 7 of the 12 indices. A negative correlation implies that as training performance increases, evaluation performance decreases. The rest of the indices had high *P-values* telling us the variables are not even correlated. Shrinkage results from Table 5.4 also imply a high degree of over-fitting from the intense levels of performance degradation. The worst performing indices during evaluation (DAX, DJIA, FTSE100, NASDAQ100 and, SMI30) also display the highest shrinkage values.

Not all markets offered the same performance. Some markets proved harder than others, specially the FTSE100 index, which had the lowest $\mathcal{SR}$ during both training and evaluation. At the other extreme we have the NASDAQ Composite where the GA found solutions with very high $\mathcal{SR}$ whose performance was also quite high during evaluation. We must note that the NASDAQ Composite was also the index offering the highest correlation between training and evaluation. The performance for this index in particular, worsened significantly for the rest of experiments.

UNIVERSIDAD DE MÁLAGA

Only a few indices offered good results during evaluation, such as the IBEX35, MIBTEL, and NASDAQ Composite. These three indices also display the highest correlation and lowest shrinkage.

	Experiment 1		Experiment 2		Experiment 3		Experiment 4	
Index	ρ	*P-val*	ρ	*P-val*	ρ	*P-val*	ρ	*P-val*
AEX25	-0.0645	0.7395	0.2045	0.0402	0.1839	0.0604	0.2739	0.0016
CAC40	-0.1804	0.3385	0.1803	0.0872	0.3889	0.0003	0.4871	0.0000
DAX	-0.2085	0.2678	-0.1476	0.1123	0.1143	0.2240	-0.0951	0.2521
DJIA	0.1453	0.4421	0.0101	0.9150	-0.3600	0.0002	-0.0409	0.6142
FTSE100	-0.2052	0.3361	-0.0832	0.5348	-0.5023	0.0001	0.2292	0.0685
IBEX35	0.1600	0.3969	0.7595	0.0000	0.5592	0.0000	0.7357	0.0000
MIBTEL	0.4024	0.0283	0.7793	0.0000	0.5260	0.0000	0.6500	0.0000
NAS100	-0.1769	0.3483	-0.1186	0.2028	-0.2061	0.0943	0.0490	0.5628
NASCOMP	0.8687	0.0000	0.2260	0.0135	0.8270	0.0000	0.1577	0.0457
SMI30	0.0140	0.9420	0.4274	0.0001	-0.2989	0.0067	0.3773	0.0001
SP500	-0.0968	0.6097	0.1747	0.0596	0.3428	0.0007	0.0292	0.7187
STOXX50	-0.0861	0.6499	0.2055	0.0403	0.0408	0.7592	0.4640	0.0000

Table 5.3: Spearman correlation coefficients between training and evaluation $\mathcal{SR}$.

	Experiment 1		Experiment 2		Experiment 3		Experiment 4	
Index	μ	σ	μ	σ	μ	σ	μ	σ
AEX25	0.9500	0.1490	0.2927	4.6145	0.4423	1.3960	0.2776	7.5724
CAC40	0.6256	0.2534	0.3540	5.5188	0.3674	3.2678	1.2282	8.7353
DAX	0.9774	0.1452	0.7622	2.5724	0.8625	1.0393	4.2722	41.0781
DJIA	1.0291	0.1355	0.7655	9.9762	0.9657	0.9248	-0.0060	19.8992
FTSE100	1.2609	0.7201	1.5734	15.9778	1.4733	1.6365	0.1477	7.5918
IBEX35	0.7352	0.1463	0.2047	2.0259	0.4850	1.1851	-0.0254	3.5151
MIBTEL	0.7113	0.1917	0.6404	2.7808	0.9452	0.4716	0.7319	3.6471
NAS100	1.0312	0.1824	-1.5656	43.0632	1.3380	1.9389	1.0641	2.0806
NASCOMP	0.7036	0.1362	0.8930	1.1509	0.5966	1.6261	1.0746	2.2786
SMI30	1.2947	0.2008	-0.5863	20.3884	1.0922	0.8213	0.8836	11.7050
SP500	0.8882	0.1810	0.7645	3.4464	1.2215	4.3731	1.2312	3.5725
STOXX50	0.9011	0.1896	-18.7088	227.3336	1.3517	15.2554	-1.1811	24.4009

Table 5.4: Mean and standard deviation of shrinkage between training and evaluation $\mathcal{SR}$.

5.6.1.1 Experiment 2: Robust Multi-Objective Optimization of $\mathcal{SR}$

In this second experiment, we transform the previous mono-objective optimization into a robust multi-objective problem. We evaluate the same strategy parameters in all 12 indices, and use the mean and standard deviation of the sortino ratio from the individual ratios obtained in each market as the robust fitness functions to optimize. The motivation behind this experiment is to analyze if our proposed robust method can help in producing better solutions than the ones provided by the last experiment. A total of 30 executions were performed.

UNIVERSIDAD DE MÁLAGA

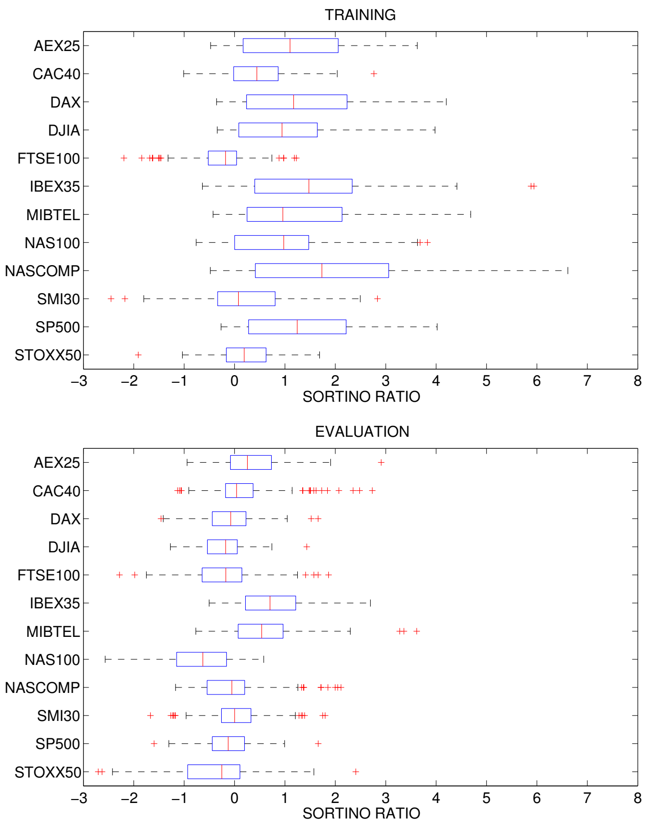


Figure 5.4: Experiment 2. Robust Multi-Objective Optimization of $\mathcal{SR}$. Results for training and evaluation.

Looking at Figure 5.4 we can see the results during training were not as high as in single-market optimization (Figure 5.3). This was expected, as we are searching for the best average solution across all indices, instead of a specific solution for just one market.

In Figure 5.5 we show a comparison between results for the evaluation phase obtained by the single-market optimization of $\mathcal{SR}$ (Experiment 1) in red, and the robust multi-objective optimization of $\mathcal{SR}$ in blue (Experiment 2). These are the evaluation results previously shown in Figures 5.3 and 5.4 which have been plotted in the same graph for easy comparison.

The robust solutions when tested in evaluation, offered, on average, similar results on most indices with regards to Experiment 1. We have to take into consideration that while in the first experiment we consider only 30 solutions which have been optimized to maximize $\mathcal{SR}$, in multi-objective optimization we have a front of non-dominated solutions showing the compromise between objectives, hence we can have solutions with lower mean $\mathcal{SR}$, but also with lower standard deviation across all markets. We evaluate

UNIVERSIDAD DE MÁLAGA

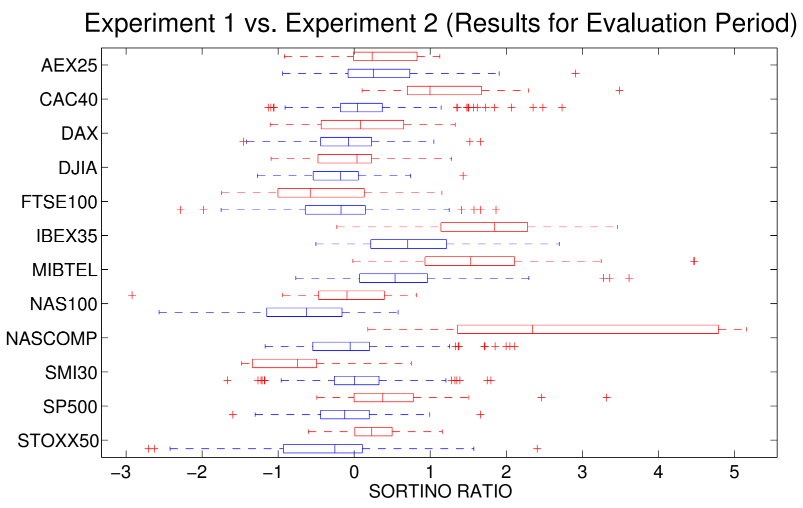


Figure 5.5: Single-Market Optimization of $\mathcal{SR}$ (Experiment 1) vs. Robust Multi-Objective Optimization of $\mathcal{SR}$ (Experiment 2). Experiment 1 evaluation results are represented in red, while Experiment 2 results are represented in blue.

all the solutions from the 30 fronts, in total 140 solutions which correspond to 4.66 solutions on average per front. We also performed a KS test to check whether the results of this experiment had a normal distribution. We can reject at the 95% confidence level that they follow a normal distribution.

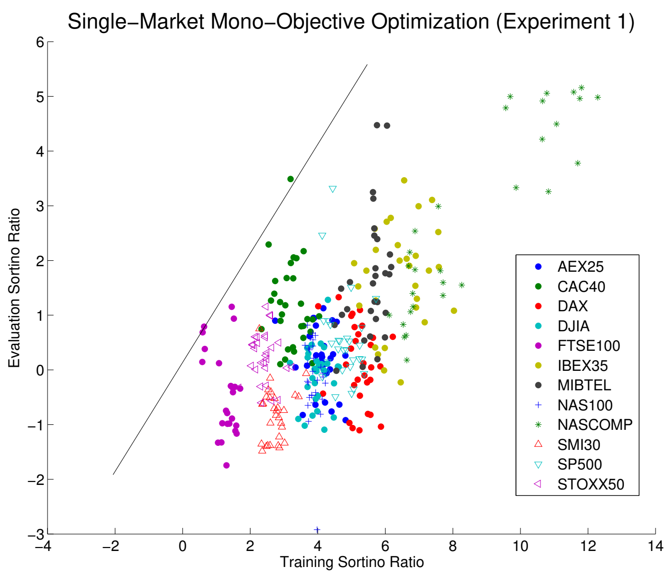
Considering all the solutions can skew to the downside the results, as solutions with lower mean $\mathcal{SR}$ are considered. This is why comparing average results can be somewhat deceiving, and we should look at the total area covered by the solutions in the box plots. If we consider the best solutions offered by both methods we can see that for some markets (AEX25, DAX, DJIA, FTSE100, SMI30 and STOXX50) the robust method was able to find better solutions. The CAC40 index shows a lot of outliers at the positive extreme.

This results show that the robust method also produced worse solutions for all indices, which can be caused by including all the solutions from every Pareto front.

We plotted each of the solutions generated during Experiment 1 and Experiment 2 in order to see the relationship between the performance of both training and evaluation phases. Figure 5.6 plots training results on the x axis and evaluation results on the y axis. This figure visually represents the performance relationship between training and evaluation phases. A line has been drawn at $x = y$ representing the ideal optimization results without shrinkage. This would be a solution that has the same performance during training and evaluation phases. Ideally, robust solutions should lie as near

UNIVERSIDAD DE MÁLAGA

as possible to the path drawn by the line.



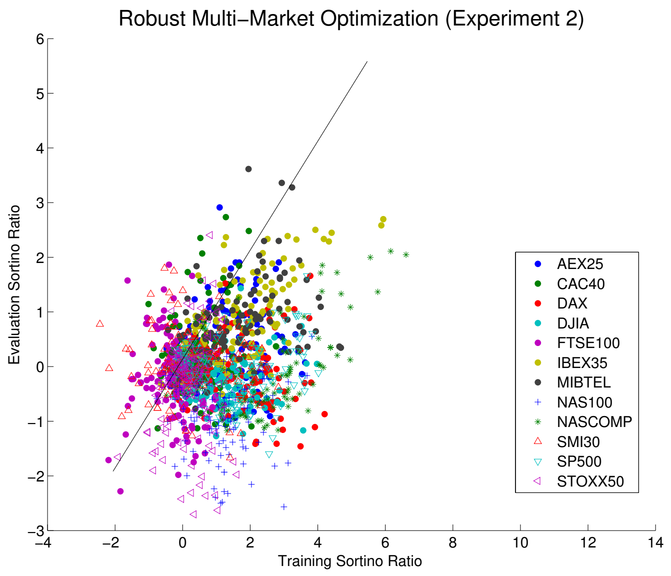


Figure 5.6: Single-Market Optimization of $\mathcal{SR}$ (Experiment 1) vs. Robust Multi-Objective Optimization of $\mathcal{SR}$ (Experiment 2). Training and evaluation results are plotted on each axis to view the relationship.

Looking at Figure 5.6, we can clearly see that the robust solutions are nearer to the line, showing a more stable performance in both training and evaluation phases. A smaller shrinking factor is also evident in Table 5.4 for most indices when compared to Experiment 1. There are two exceptions, the

UNIVERSIDAD DE MÁLAGA

FTSE100 and NASDAQ Composite indices, which offered higher shrinkage in Experiment 2, while the NASDAQ 100 and STOXX50 indices had negative shrinkage, meaning higher performance during evaluation than training. In Table 5.3 we can also see that the correlation between training and evaluation for this experiment is higher for most indices. Some indices (AEX25, CAC40, SP500 and STOXX50) had negative correlation for Experiment 1, evidencing over-fitting, that turned to positive correlations in Experiment 2. The other two remaining negative correlations from Experiment 1, the DAX and NASDAQ100 indices, also had negative correlation but to a lesser degree. The best performing indices during evaluation for this experiment were the IBEX35 and MIBTEL. These two indices were also the ones displaying the highest correlations in Table 5.3.

Overall, the results for this experiment show that even though the performance during evaluation might not be better (on average) to that of Experiment 1, our robust method has produced solutions that are more stable and robust, as evidenced by the higher correlations and lower shrinkage.

5.6.1.2 Experiment 3: Single-Market Optimization of both $\mathcal{SR}$ and $\mathcal{MDD}$

In this experiment, the fitness functions previously mentioned in Section 3.4, the sortino ration and maximum drawdown ($\mathcal{SR}, \mathcal{MDD}$) are optimized for each market. The motivation for this experiment is showing how incorporating the $\mathcal{MDD}$ as an additional objective to optimize, leads to better solutions. This experiment also serves as the basis of comparison with the next experiment, where we transform it into a robust multi-market optimization with four objectives.

30 executions are conducted for every index. Every execution produced a unique Pareto front with on average 4.3 solutions, resulting in a total of 131 solutions showing the compromise between $\mathcal{SR}$ and $\mathcal{MDD}$. We performed a KS test on the results obtained with this experiment, which revealed they do not follow a normal distribution with a 95% confidence level.

Looking at Figure 5.7 we can see that results achieved in terms of training $\mathcal{SR}$ were in line with those obtained in the single-market optimization of $\mathcal{SR}$ (Experiment 1), arriving at solutions with high $\mathcal{SR}$ for the in-sample data.

Using $\mathcal{MDD}$ as an additional objective function, has improved the quality of solutions (regarding this objective), as compared to the first single-market mono-objective experiment. In the first experiment the solutions generated only considered $\mathcal{SR}$. Figures 5.8 and 5.9 show a significant reduction of $\mathcal{MDD}$ in the strategies generated for this experiment, evidencing the benefits of including this objective.

We will compare these results in more detail in the next section together with the ones obtained from the next robust experiment.

UNIVERSIDAD DE MÁLAGA

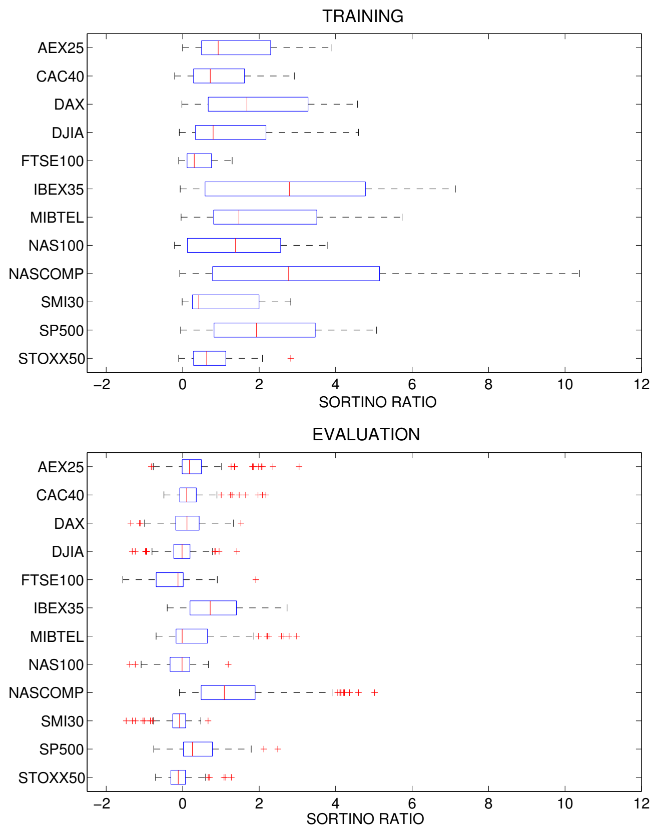


Figure 5.7: Experiment 3. Single-Market Optimization of both $\mathcal{SR}$ and $\mathcal{MDD}$. Results for training and evaluation.

5.6.1.3 Experiment 4: Robust Multi-Objective Optimization of both $\mathcal{SR}$ and $\mathcal{MDD}$

The motivation behind this experiment is showing how our robust methodology can be applied in problems where there are more than two objectives. In the previous experiment we have demonstrated how using $\mathcal{MDD}$ as an additional objective can prove useful. For this experiment, the previous objectives from Experiment 3, $(\mathcal{SR}, \mathcal{MDD})$ are transformed into four objective functions: $(\mathcal{SR}_\mu, \mathcal{SR}_\sigma, \mathcal{MDD}_\mu, \mathcal{MDD}_\sigma)$. The optimization is performed across all markets instead of individually for each market. We run 30 independent executions, where we test the same parameter settings over all the markets. At the end of all runs we have a total of 191 solutions. On average each of the 30 Pareto fronts had 6.36 solutions. We conducted a KS test on the results obtained with this experiment, which revealed they do not follow a normal distribution with a 95% confidence level.

We have provided Figure 5.10 showing a box plot comparing training and evaluation results for this experiment. We can see that training performance

UNIVERSIDAD DE MÁLAGA

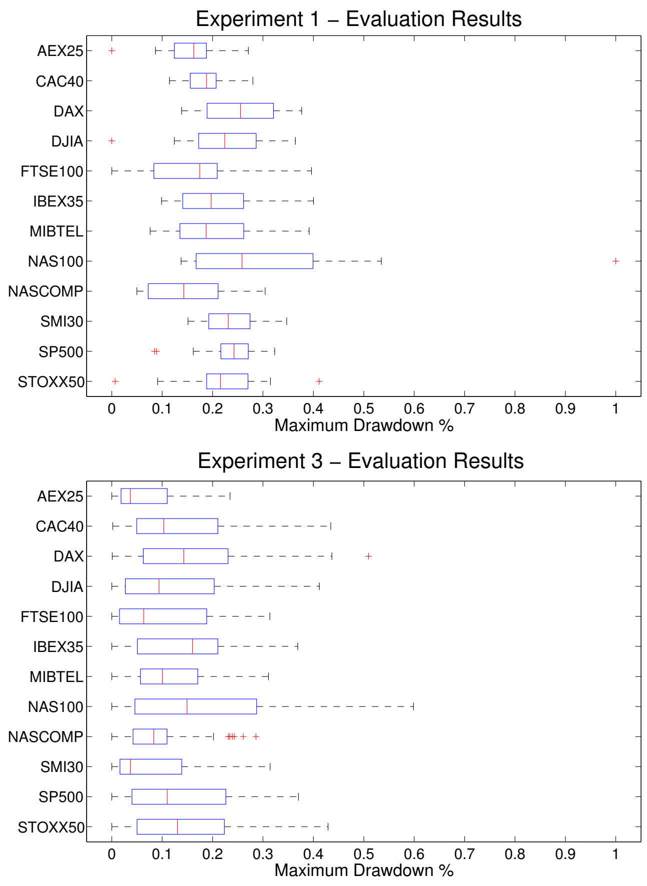


Figure 5.8: Single-Market Optimization of $\mathcal{SR}$ vs. Single-Market Optimization of both $\mathcal{SR}$ and $\mathcal{MDD}$, (Experiments 1 vs. 3). Benefits of using $\mathcal{MDD}$ as an objective.

is much less than in Experiment 3, as we are looking for the best average strategy across several markets. This is consistent to what happened in Experiments 1 and 2.

In Figure 5.11, we compare the evaluation results offered by this experiment with the previous experiment. This figure is not as conclusive for asserting the robust method offered better results, but it can be appreciated that for most of the indices, there have been better solutions generated at the right end, seen as outliers in the plot (red + sign). Some indices have shown worse results such as the NASDAQ100 and STOXX50, which yielded a lot of bad solutions, but if we compare them by the best solutions, we can see that the robust method was capable of providing better solutions for 8 of the 12 indices.

Looking at $\mathcal{SR}$ performance during evaluation tells us only part of the story. As we noted earlier, erratic behaviour between training a evaluation

UNIVERSIDAD DE MÁLAGA

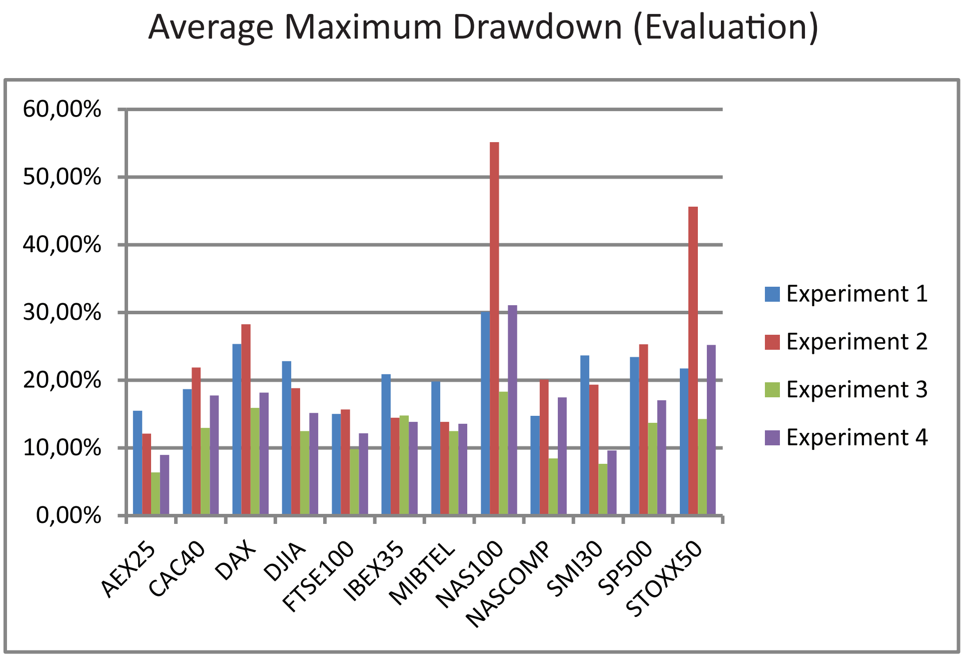


Figure 5.9: Reduction in $\mathcal{MDD}$ obtained by including it as an objective to be optimized. Experiments 1 and 2 were only optimzed w.r.t. $\mathcal{SR}$.

performance is indicative of poor solutions that have been over-fitted. We are interested in producing solutions displaying stable performance when tested out-of-sample. These solutions should have a lower shrinkage and higher correlations between training and evaluation.

We have plotted the relationship between training and evaluation in Figure 5.12 and compared the results to the previous Experiment 3. This plot makes visible the higher correlation and lower shrinkage obtained with this experiment. We can see how the solutions lie closer to the line. The plot also shows a lot of solutions that are on the top side of the line. These are solutions whose performance during evaluation is higher than training. We must note that even though this would not be desirable under single-market optimization, in robust multi-market optimization can mean perfectly valid solutions. If during single-market optimization there is a big deviation between both training and evaluation, especially when evaluation results are high after poor training performance, one might doubt the validity of those results, as during the learning phase the same performance was unattainable. In robust optimization, the solution has seen more market scenarios, whereas in single-market optimization it has only seen one. It could perfectly happen that although the parameters have scored badly for a specific market, they might have performed better in others, thus we should consider those solutions, even when there is a big discrepancy with training results.

UNIVERSIDAD DE MÁLAGA

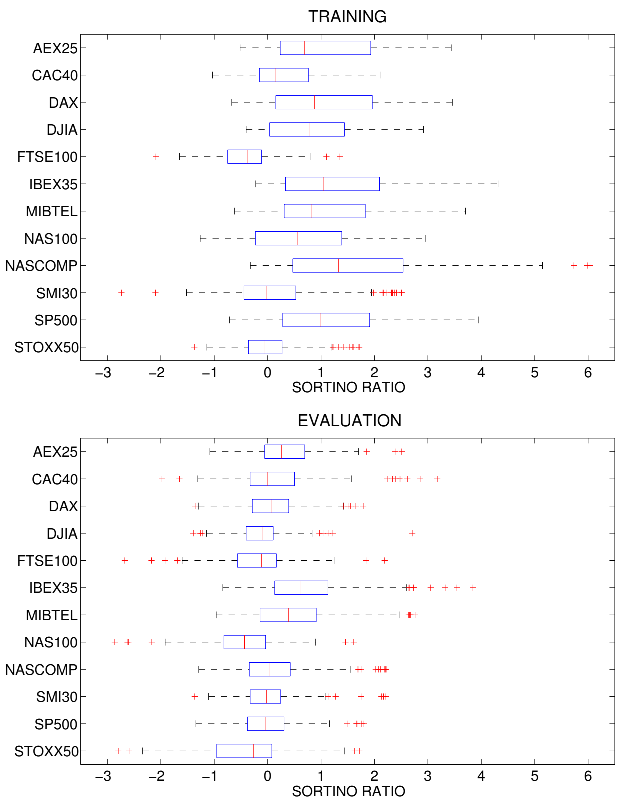


Figure 5.10: Experiment 4. Robust Multi-Objective Optimization of both $\mathcal{SR}$ and $\mathcal{MDD}$. Results for training and evaluation.

In Figures 5.10 and 5.12 can also be seen a lot of negative scores during training, while in single-market optimization there are not any. This is normal and expected behaviour as in single-market optimization the GA will find a good scoring solution for that particular market, while in robust multi-market, as we are looking for the best average strategy across all markets, it can happen that some solutions are produced whose performance during training for a specific market is not good. This negative scores will certainly skew the results when compared to single-market optimization results, which did not have negative solutions.

Looking at Table 5.3 we can see that the correlations between training and evaluation $\mathcal{SR}$ are higher for 9 of the 12 indices when compared to Experiment 3. The remaining 3 indices (DAX, NASDAQ Composite, and SP500) offered lower correlations for this experiment. Three of indices that had negative correlations in Experiment 3 (FTSE100, NASDAQ100 and SMI30) turned positive for Experiment 4. These three indices also offered the poorest evaluation results w.r.t. $\mathcal{SR}$. Table 5.4 shows shrinkage has been reduced for 8 of the 12 indices. We can also see three indices with negative

UNIVERSIDAD DE MÁLAGA

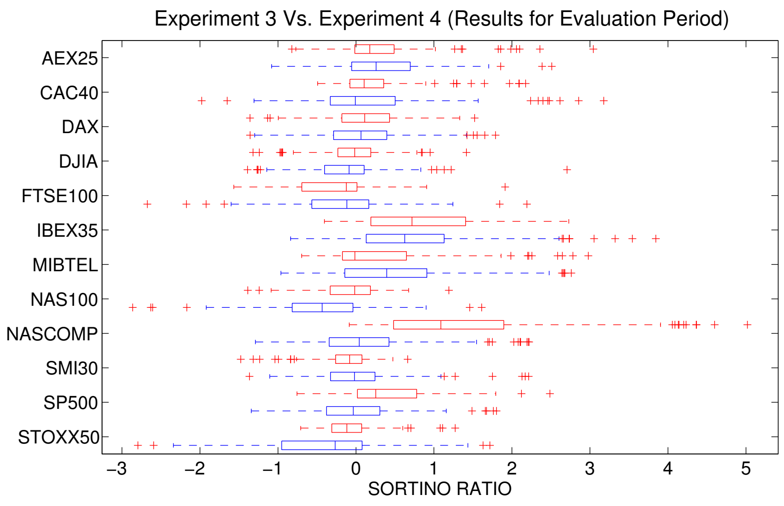


Figure 5.11: Single-Market Optimization of both $\mathcal{SR}$ and $\mathcal{MDD}$ (Experiment 3) vs. Robust Multi-Objective Optimization of both $\mathcal{SR}$ and $\mathcal{MDD}$ (Experiment 4). Experiment 3 evaluation results are represented in red, while Experiment 4 are represented in blue.

shrinkage (DJIA, IBEX35 and STOXX50), this is caused by solutions whose performance in evaluation is greater than training. All this negative shrinking values can be seen as the solutions on top of the line of Figure 5.12. A side effect of this can be seen as an increase in the standard deviation from the mean shrinkage.

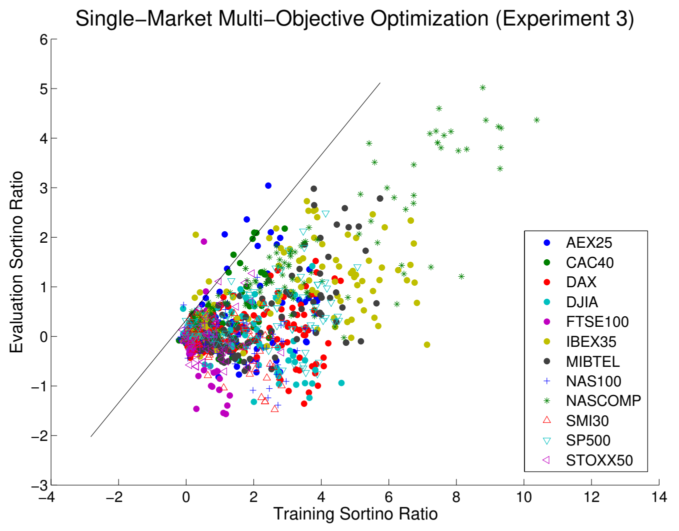
Once again the robust multi-market optimization results look more stable and improve the best $\mathcal{SR}$ obtained during evaluation for most indices as compared to single-market optimization results.

If we consider $\mathcal{MDD}$, comparing the results of Figure 5.9 between Experiment 2 and Experiment 4 we can see that there has been a reduction of $\mathcal{MDD}$ and the benefit of including this objective, as it yielded strategies with reduced loses. The results show that using this objective has led to improved solutions in terms of reduced risk. We bring to attention that this bar chart also shows an increase in $\mathcal{MDD}$ for this experiment when compared to previous Experiment 3. We can attribute this increase in $\mathcal{MDD}$ to robust solutions being more general, and this loss of specificity to a particular market can lead to worse performance in terms of $\mathcal{MDD}$.

Figure 5.13 provides a comparison of the average profits in percentage terms obtained by the 30 best solutions (w.r.t. $\mathcal{SR}$) of every experiment. The reason for filtering results this way is to compare results between the first experiment (mono-objective $\mathcal{SR}$) with the rest of multi-objective optimizations.

We have annualized the total return dividing by the number of years in

UNIVERSIDAD DE MÁLAGA



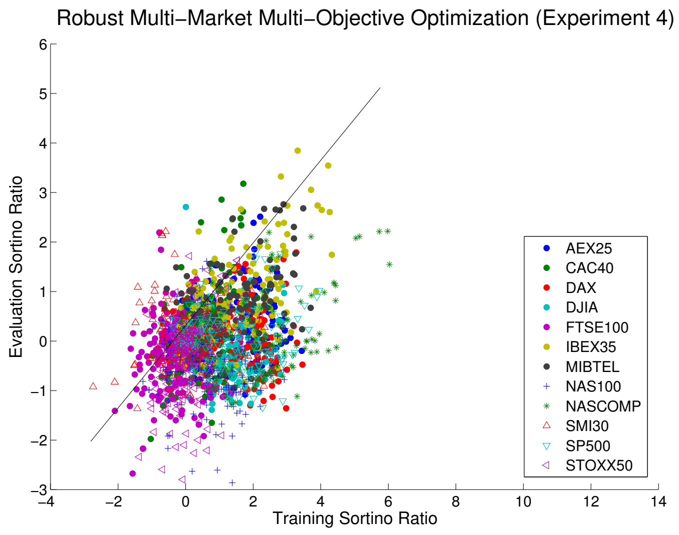


Figure 5.12: Single-Market Optimization of both $\mathcal{SR}$ and $\mathcal{MDD}$ vs. Robust Multi-Objective Optimization of both $\mathcal{SR}$ and $\mathcal{MDD}$. Training and evaluation results are plotted on each axis to view the relationship.

the dataset. This is done in order to properly compare between training and evaluation, as their period lengths are different. We also provide the profit obtained by the passive strategy *"Buy and Hold"*, which is a reflection of the returns obtained by the market.

As we can see most of the solutions found in all experiments were able to substantially beat buy and hold during evaluation. The exceptions are the Dow Jones Industrial Average, Nasdaq 100, and SP500 indices. This indices had exceptional *bull markets* where it is very hard to beat buy and

UNIVERSIDAD DE MÁLAGA

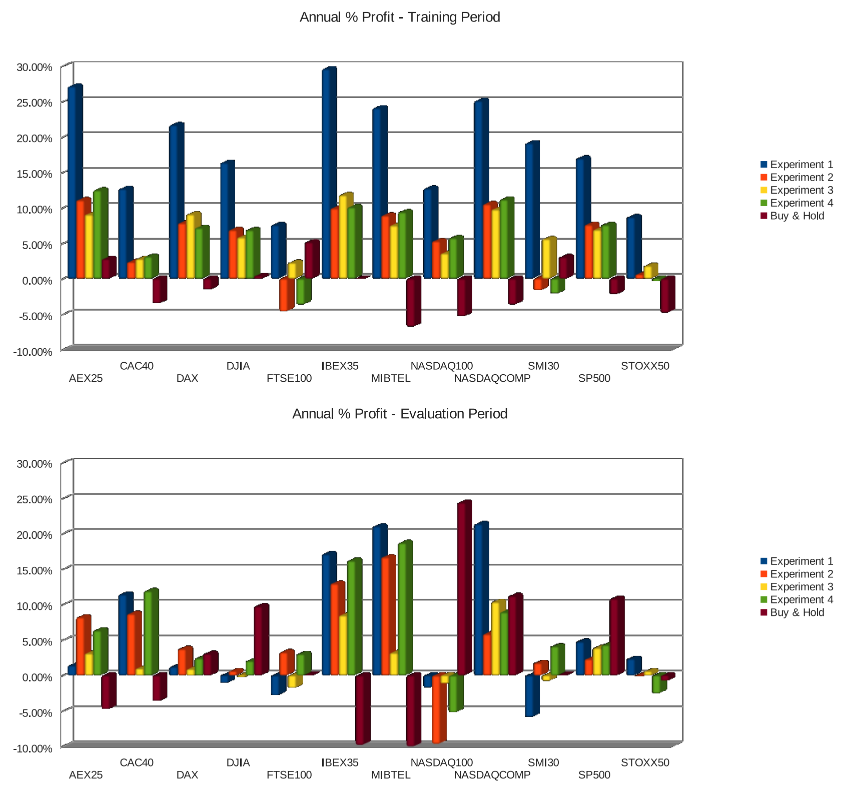


Figure 5.13: Average annual % profit during training and evaluation compared to *"Buy & Hold"*

hold. The interesting exception was the Nasdaq Composite index which offered exceptional results in experiment 1 almost doubling buy and hold performance while experiments 3 and 4 were slightly below.

The best evaluation results were obtained in the Spanish IBEX35 and the Italian MIBTEL, which happened to experience acute bear markets during evaluation. The percentage results almost tripled buy and hold (-10%).

The percentage returns are consistent with the sortino ratios obtained. Higher sortino ratios lead to higher returns while negative sortino ratios have negative returns.

5.7 Conclusions

In this chapter we have presented a robust multi-objective optimization framework which considers increasing the environmental conditions whereby the search is conducted over multiple markets with the goal of finding robust solutions which have not been over-fitted.

Our proposed robust methodology has improved results during evaluation

UNIVERSIDAD DE MÁLAGA

for the majority of indices considered in this chapter, as shown by the higher correlations and less shrinkage of solutions that have been optimized over multiple markets.

Our work offers evidence that the optimization over a wide variety of similar financial instruments can help in producing robust solutions whose performance is not significantly degraded when they are tested out-of-sample, or market conditions change. Our proposed robust method has produced significantly more solutions whose performance is near the optimal 1 to 1 relationship between training and evaluation, than those offered by a single-market optimization.

We also offer evidence that incorporating the maximum drawdown as an objective, leads to better solutions, as the GA has a way of seeing more details about the performance of a strategy, resulting in strategies that trade better with less risk.

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA

Chapter 6

Robust technical trading strategy discovery

I love fools' experiments.
I am always making them.
Charles Darwin

6.1 Introduction

Last chapter we saw how to optimize robustly the parameters of a TTS. In this chapter we plan to explore how to develop from "scratch" a robust TTS using GP (*Genetic Programming*), that will be used to manage a portfolio of stocks from the Spanish market. The investigated method is used to determine potential buy and sell conditions for stocks, aiming to yield robust solutions able to withstand extreme market conditions, while producing high returns at a minimal risk.

In this chapter we face again the recurring problem of over-fitting, maybe even aggravated as a consequence of using GP, as the size of the search space has increased compared to last chapter. In order to solve it, we explore a random sampling method we call RSFGP which instead of calculating the fitness over the whole dataset, calculates it on randomly selected segments. This method shows improved robustness and out-of-sample results compared to SGP and a VAFGP.

In this chapter we explore how we can design solutions that display similar performance for both in-sample and out-of-sample data, as well as solutions that can resist abrupt trend changes and extreme volatility periods. Our approach is substantially different to previous work and is centered around how we calculate the fitness function. Our main contribution lies in evaluating the fitness using a random sampling method which will explain later in Section 6.4.1.

UNIVERSIDAD DE MÁLAGA

Consequently, in this chapter a robust GP evolutionary approach will be presented to automate buying and selling decisions in order to maximize the Sterling ratio (total return divided by maximum drawdown). The proposed method will be tested on a basket of 21 stocks from the Spanish market using 13 years of daily price data and compared to the IBEX35 market index, the results will be analyzed and some possible conclusions will be discussed.

These strategies are evaluated using 21 of the most liquid stocks of the Spanish market. The achieved results clearly outperform *Buy & Hold*, SGP and VAFGP. Additionally, the solutions obtained with the training data during the experiments clearly show during testing robustness to steep market declines as seen during the European sovereign debt crisis experienced recently in Spain. The solutions learned were able to operate for prolonged periods, which demonstrated the validity and robustness of the rules learned, which are able to operate continuously and with minimal human intervention.

To sum up, the method developed in this chapter is able to evolve TTSs suitable for all market conditions with promising results, which suggests great potential in the method generalization capabilities. The use of additional financial metrics alongside popular *technical indicators* enables the system to increase the stock return while proving resilient through time. The RSFGP system is able to cope with different types of markets achieving a portfolio return of 31.81% for the testing period 2009 to 2013 in the Spanish market, having the IBEX35 index returned 2.67% during the same period.

6.2 Literature review

GP was first employed by Allen and Karjalainen (1999) for technical trading rule discovery. The dataset used in their experiments was the S&P 500 index using daily prices from 1928 to 1995. Their results demonstrated that although GP could find profitable trading rules, it failed to produce excess-returns over the passive strategy of *Buy & Hold*, which consists in buying on the first evaluation day and selling on the last.

Neely (2003) extends the previous work by Allen and Karjalainen (1999) using a risk adjustment selection criterion to generate rules with the hope of improving performance. However, the results show no evidence that the rules significantly outperform *Buy & Hold* on a risk-adjusted basis.

Becker and Seshadri (2003) present results of GP-evolved technical trading rules, which outperform a buy-and-hold strategy on the S&P 500 after taking into account transaction costs. They introduce several changes to the original work of Allen and Karjalainen (1999), which include a complexity-penalizing factor, a fitness function that considers consistency of performance, and co-evolution of separate buy and sell rules. Monthly data is used instead of daily.

UNIVERSIDAD DE MÁLAGA

Lohpetch and Corne (2009) replicate the work of Becker and Seshadri (2003) and the authors find that the results are sensitive to the data periods chosen for the experiments. Their results are improved by using a validation set, used for choosing the best rule found during training.

Mallick and Lee (2008) used GP to find trading rules on the thirty component stocks of the Dow Jones Industrial Average index. The authors find Statistical evidence of outperforming *Buy & Hold* in falling markets, and confirm that GP based trading rules generate a positive return under bull (rising) and bear (falling) markets.

Yan and Clack (2010) use GP for building a symbolic regression expression that measures the attractiveness of each stock; Each month a portfolio is constructed with the most attractive stocks according to the GP model. The portfolio is a market neutral long/short portfolio of Malaysian equities. The authors propose two approaches for evolving robust trading rules. First by splitting the training dataset into three extreme environment periods: up, down and sideways volatile. Secondly instead of using just one solution, a voting comity is used, formed by the three best solutions trained on each of the extreme environments. The authors show results that considerably beat the benchmark index, but the results have a significant caveat, i.e. they used a small out-of-sample period (July 1997 to December 1998), which is before the training period (January 1999 to December 2004). Monthly data was used to simulate portfolio, meaning at the beginning of a month the stocks which the system recommends are bought, and at the end of the month the position is reassessed.

Hsu (2011) use a hybrid Self-Organizing Map (SOM) GP system where the SOM unsupervised neural network is used to cluster the time series into similar segments. This segments are then used by the GP system to learn trading rules for each of the market conditions detected by the SOM. During testing the SOM is used to classify the unseen time series and select the best GP solution found during training on similar time series. Our method has some advantages over this method, as it does not require the use of any unsupervised method to group similar time series segments and thus is less computationally expensive. Another important advantage of our method is that it provides a single robust solution that works well in all market conditions.

Mousavi et al. (2014) use a dynamic GP portfolio trading system based on technical indicators. The authors extend the classical GP algorithm to a multi-tree GP forest that is able to extract multiple trading rules, one for each of the assets considered. Since the traditional GP structure is not able to cope with this specific problem, the consequent parts of the rules are designed as a crisp function of the weights of the stocks in the portfolio. The fitness measure employed in this study is the conditional Sharpe ratio, a modification of the original Sharpe ratio, using CVAR (Conditional Value at

UNIVERSIDAD DE MÁLAGA

Risk) as the divisor instead of the standard deviation of returns. The system was trained and tested on 15 stocks from the Iranian Stock Exchange and 15 stocks from the Toronto Stock Exchange with a sliding window approach using 4.5 years of daily stock prices. Our method has some advantages, firstly being simpler, as a single rule is evolve to trade all assets in the portfolio, it is less computationally expensive, and secondly evolving a single trading rule that can be applied to all the stocks increases the robustness of solutions. Thirdly our method has been trained on 8 years of data and tested during the next 5 years, proving its robustness over an extended period of time.

Gypteau et al. (2015) use an intrinsic time scale based on DC (directional changes) combined with Genetic Programming to find an optimal trading strategy that forecasts future price moves. A DC event is identified by a change in the price of a given stock greater than a predefined threshold value, which was in advance decided by the user. The authors use a total return as the fitness measure and use two stocks from the UK market and the NASDAQ and NYSE indices to evolve their solutions over a period of 1000 days for training and 500 days for testing. Their results showed that the strategies evolved by the GP are more profitable when using multiple threshold values than using a fixed threshold value, providing evidence that DC can be used for forecasting and that combining multiple thresholds is beneficial. In comparison to our proposed method, this method only uses DC and does not consider other technical or financial metrics. It is tested on a very limited selection of assets (2 stocks and 2 indices), and does not compare the results obtained with other traditional strategies such as Buy & Hold, making the results obtained in the study hard to interpret.

Luengo et al. (2015) manually divide the stock price time series into 3 segments of 4 years which sometimes overlap, and then this previous segments are again divided into 3 different period, 1 for pre calculating the technical indices, 2 for GP training and 3 for testing. In comparison our approach has some advantages as we can use the best evolved rule in all market conditions proving to be resilient to regime switches in the data. It also has the advantage of not requiring human intervention in manually dividing the time series into distinct market regimes and providing a robust solution that produces good results in all market environments.

In this chapter, we use a similar approach to Yan and Clack (2010), as we think that *which* data and *how* it is presented is crucial for any machine learning to occur; after all you can only learn what's on the data. But our approach substantially differs as we use a random sampling method at the GP individual level instead of hand-picking different bull, bear and volatile scenarios for training. Secondly we treat the problem as classification problem instead of symbolic regression. Our GP expression returns a boolean value that we interpret as a trading signal.

One of the main advantages of our proposed method over Yan and Clack

UNIVERSIDAD DE MÁLAGA

(2010), is that our random sampling method does not require any user intervention in order to divide the stock price time series. Another important advantage of randomly sampling the time series is increasing the robustness of the solutions evolved. Lastly, our method uses daily data instead on monthly data, hence it is quicker to react to abrupt changes in market conditions. One possible weakness of our method in comparison with the references previously discussed, is that it is more computationally expensive, as the fitness has to be calculated over more data than other methods who only use the whole time series for rule discovery, or methods that divide the time series into bullish, bearish and sideways trends, as randomly sampling the time series produces some overlap between segments.

6.3 Problem description

We are interested in exploring how to evolve robust GP solutions and which techniques we can employ to steer away solutions from being over-fitted. Ideally solutions should be robust and present similar performance for both in-sample and out-of-sample datasets.

We approach the problem as a binary classification problem. A GP rule is evolved using the functions and terminals provided and evaluates to a boolean that we interpret as a buy or sell signal.

6.3.1 Portfolio simulation

To evaluate the GP evolved rules we simulate a long only portfolio of the 21 largest and most liquid Spanish stocks. We choose the Spanish market as the testing period from 2009 to 2013 has been particularly volatile, specially in the summer of 2012 with the outbreak of the sovereign debt crisis in Europe. We use as the reference benchmark the IBEX35 index.

Portfolio returns are calculated in the following manner; at the initial evaluation period, the portfolio starts with W_0 in cash. At every day the GP rule is evaluated and the stocks which are classified as True are bought (or maintained if already in the portfolio), and those classified as False are sold if owned in the portfolio. The total portfolio value W_t is calculated daily by valuing the shares at the closing price of each day plus the value of the cash account. We do not use leverage or reinvest profits and each purchase is allocated a fixed amount of cash of 10,000.00 €. Transaction costs of 0.3% are included in the calculations.

6.3.2 Fitness function

We employ a single objective approach to determine the quality of the evolved solutions. We employ risk-adjusted metric, the Sterling ratio (see

UNIVERSIDAD DE MÁLAGA

Chapter 4.5.4) to measure the fitness of individuals. The Sterling ratio is used in Dempster and Jones (2001) and Zhang and Ren (2010) as the fitness measure in their systems and is calculated as the total return divided by the maximum drawdown. As we saw in last chapter, including the maximum drawdown is beneficial, even when other metrics, that consider risk are present. In the problem addressed in this chapter we are interested in generating a single best solution instead of a Pareto Front of non-dominated solutions. This is the justification behind using the Sterling ratio instead of the Sortino ratio for the fitness of individuals.

6.4 Algorithmic approach

We employ an elitist $\mu + \lambda$ evolutionary process where from μ parents we generate λ offspring and the best individuals from both μ and λ form the population of the next generation. $\mu + \lambda$ has the advantage of not losing the best solutions during evolution as they are never replaced by inferior individuals.

We utilize strongly-typed GP which allows for the declaration of data types of functions and terminals, and offers the advantage of limiting the search-space to syntactically valid expressions only.

Figure 2.3 shows an example of a trading rule generated by GP.

6.4.1 *Random Sampling Fitness Genetic Programming*

We are concerned with endowing generalization and environmental robustness to the solutions evolved by GP. We achieve this by exposing the individuals to random market situations sampled from the original dataset.

Instead of evaluating the fitness of the individual in the whole training dataset, our approach consists in selecting n random segments $\{s_1, s_2, \ldots, s_n\} \in S$ from the whole training dataset S and calculating the fitness on each of the segments. Assuming I_i is an individual in the population of solutions, and $f_{I_i}^{s_j}$ is the fitness of the individual I_i on segment s_j The final fitness $\overline{F}_i$ is calculated as the mean fitness obtained in the n randomly sampled segments of S. The random sampling is done with replacement at the individual level, i.e. each individual is always evaluated on different randomly selected segments of the original dataset.

$$\overline{F}_i = \frac{1}{n} \sum_{j=1}^{n} f_{I_i}^{s_j} \tag{6.1}$$

We have chosen $n = 100$ for the number of segments while the length is set at one trading year or 255 days. If the trading rule does not generate any buy or sell signals for that section, a penalty is used which sets the fitness

UNIVERSIDAD DE MÁLAGA

value to −9.99. This is done in order to penalize solutions do not trade and would have a fitness value of 0 and forces evolution to choose poor solutions with low fitness values (but with some trading activity) over solutions that did not trade.

6.4.2 *Volatility Adjusted Fitness Genetic Programming*

Yan and Clack (2010) use a volatility adjusted fitness that is quite similar to our random sampling method but using the standard deviation of the fitness as the divisor. See Equations 6.2 and 6.3. In the experimental results of section 6.6.3 we also study the effects of using a volatility adjusted fitness and compare it to our proposed random sampling method.

$$\sigma = \sqrt{\sum_{j=1}^{n} \frac{(f_{I_i}^{s_j} - \overline{F}_i)^2}{n}} \tag{6.2}$$

$$\textbf{Volatility adjusted fitness} = \frac{\overline{F}_i}{\sigma} \tag{6.3}$$

6.5 Methodology

In this section we explain our methodology approach.

6.5.1 Dataset used

Our dataset consisted of 14 years of daily prices (Open, High, Low, Close, Volume) adjusted for splits and dividends obtained from Reuters. Table 6.1 shows the Reuters symbol and company name of the stocks used. We used 21 of the largest and most liquid stocks for the Spanish Market for which we had data for the 14 years of our study. The period from January 2000 to December 2008 is used for searching for the optimal GP rule while the out-of-sample testing period used for testing starts on January 2009 and ends on December 2013. We use the IBEX35 index as the reference benchmark.

From the daily prices dataset we compute 264 company specific features, see Table 6.2. These features are used during the evolution as terminals in the GP tree. The first 200 days of the year 2000 are used to compute the initial technical indicators, hence they are not included in the training results.

6.5.2 GP parameter settings

Table 6.3 shows summary of the GP parameters and operator functions used in our experimentation. These parameters, which were decided upon after some preliminary testing, are held constant during all our experiments.

UNIVERSIDAD DE MÁLAGA

Symbol	Company Name
ABE.MC	Abertis Infraestructuras
ACS.MC	Actividades de Construcción y Servicios
ANA.MC	Acciona
BBVA.MC	Banco Bilbao Vizcaya Argentaria
BKT.MC	Bankinter
EBRO.MC	Ebro Foods
FCC.MC	Fomento de Construcciones y Contratas
GAM.MC	Gamesa Corporación Tecnológica
GAS.MC	Gas Natural SDG
IBE.MC	Iberdrola
IDR.MC	Indra Sistemas
JAZ.MC	Jazztel
MAP.MC	Mapfre
OHL.MC	Obrascon Huarte Lain
POP.MC	Banco Popular
REE.MC	Red Electrica Española
REP.MC	Repsol
SAN.MC	Banco Santander
SCYR.MC	Sacyr
TEF.MC	Telefónica
VIS.MC	Viscofán

Table 6.1: Stocks used in the experiments of this chapter

1. Return from N periods	$N = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200$
2. Simple moving average of returns (N periods)	$N = 3, 4, 5, 6, 7, 8, 9, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200$
3. Exponential moving average Close (N periods)	$N = 3, 4, 5, 6, 7, 8, 9, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200$
4. Bollinger Bands (N periods)	$N = 3, 4, 5, 6, 7, 8, 9, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200$
5. Internal Bar Strength (N periods)	$N = 0, 1, 2, 3 \ldots, 30$
6. Relative Strength Index (N periods)	$N = 5, 6, 7, \ldots, 30$
7. Volatility (N periods)	$N = 3, 4, 5, 6, 7, 8, 9, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200$
8. CAPM α (N periods)	$N = 10, 11, 12, \ldots, 60$
9. CAPM β (N periods)	$N = 10, 11, 12, \ldots, 60$
10. Sharpe ratio (N periods)	$N = 10, 13, 16, 19, \ldots, 60$

Table 6.2: Description of features in the terminal set

6.5.3 Computational environment

Our test machine consisted of a dual Intel Xeon E5-2687W @ 3.10 GHz workstation running Ubuntu 12.04.04 Linux with 64 Gb of RAM. We implemented our GP system and portfolio simulation using Python 2.7.3 and Distributed evolutionary algorithms in Python (DEAP) (Fortin et al., 2012).

6.6 Results

We execute 30 independent runs in all of our experiments, and measure the robustness of solutions using shrinkage (Mehta and Bhattacharyya, 2004; Berutich et al., 2014). Shrinkage is calculated as the percentage change in

UNIVERSIDAD DE MÁLAGA

Algorithm type	Strongly-typed $\mu + \lambda$
Population size	$1,000$
Initialization	Ramped half and half
Function set	$+, -, *, /, <, >$, and, or, if-then-else, isBetween
Terminal set	264 Company specific features (See Table 6.2)
Crossover operator	Single point crossover
Crossover fraction	80%
Mutation operator	Uniform mutation
Mutation fraction	10%
Max. initial tree depth	6
Termination	100 generations

Table 6.3: GP parameter settings

performance between training and testing data. In order to better asses the performance of the evolved strategies we include together with the sterling ratio fitness metric, the total return and the Sharpe ratio. Likewise, we analyse the mean daily returns and volatility of the portfolios generated by the various methods. Statistical analysis has been conducted on the results at the 5% significance level.

6.6.1 Standard GP

We use a standard genetic programming (SGP) approach as the basis of comparison between the random sampling fitness (RSFGP) and the volatility adjusted fitness methods (VAFGP). In SGP the fitness of the individual is calculated on the whole training dataset.

As we can see in Figures 6.1 and 6.2, SGP achieves a very high sterling ratio during training, with a mean value of 6.8398, but very poor out-of-sample results during testing with a mean sterling ratio of 0.2723. Figure 6.3 shows the shrinkage between training and testing results. SGP has the highest shrinkage evidencing over-fitting of the solutions. The testing performance of SGP solutions is degraded between -94.94% and -104.02% on average as Table 6.4 shows. We also provide a summary of the results in Table 6.5.

	Sterling	Total Return	Sharpe
SGP	-94,94%	-104,02%	-96,07%
RSFGP	-28,84%	-1,29%	13,03%
VAFGP	-56,10%	-38,12%	-25,69%

Table 6.4: Shrinkage between training and testing results.

SGP tends to over-fit solutions. 8 out of the 30 executions (26.66%) were so over-fitted that no results were produced during out-of-sample testing as

UNIVERSIDAD DE MÁLAGA

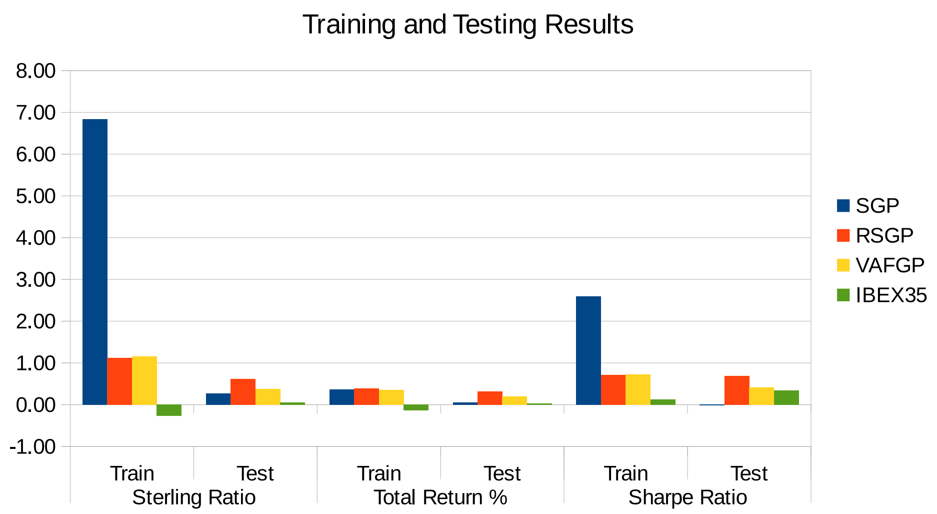


Figure 6.1: Mean results obtained for training and testing datasets.

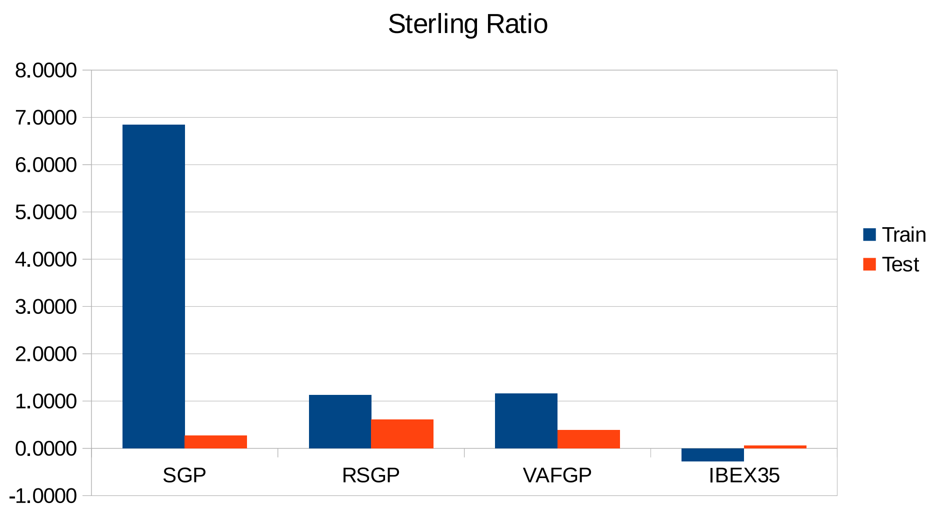


Figure 6.2: Mean sterling ratio obtained in all executions.

the rules where never triggered. Only 7 solutions out of the 30 executions (23.33%) had acceptable results when tested out-of-sample.

As an illustration we show in Figure 6.4 one of the solutions generated by SGP.

We also analyse the returns of the simulated portfolios in Figure 6.8. We can clearly see that SGP delivers the worst performance in terms of inferior mean daily returns. There is cluster of SGP portfolios whose standard deviation of returns is lower than the rest. This is due to a high percentage

UNIVERSIDAD DE MÁLAGA

	Sterling Ratio		Total Return		Sharpe Ratio	
	Train	Test	Train	Test	Train	Test
SGP	6.8398	0.2723	37.08%	5.79%	2.5877	-0.0175
RSFGP	1.1235	0.6124	38.83%	31.81%	0.7103	0.6933
VAFGP	1.1584	0.3809	35.76%	19.85%	0.7284	0.4154
IBEX35	-0.2708	0.0521	-13.65%	2.67%	0.1223	0.3389

Table 6.5: Mean results obtained in the study

of solutions that traded rarely and had a very low market exposition during out-of-sample testing. This can also be seen by looking at Figure 6.4 as we can see a very low volatility of the fund value during testing.

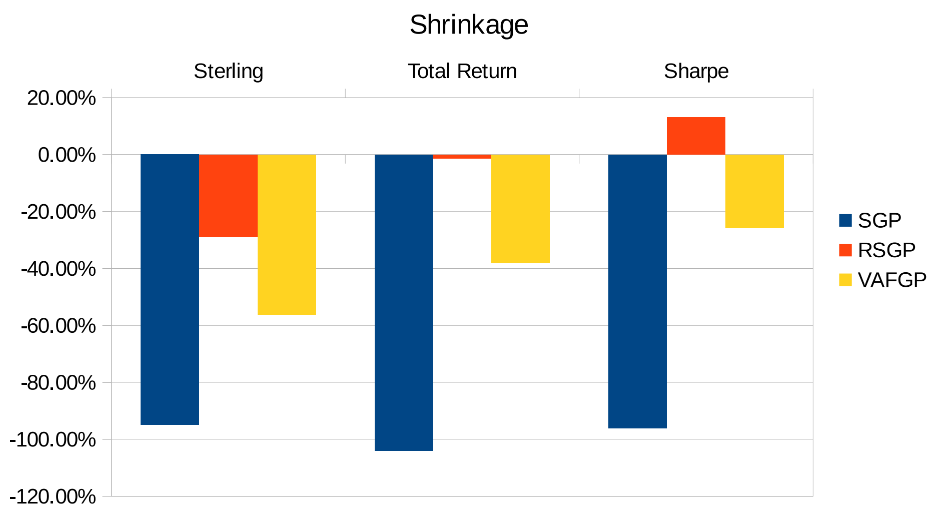


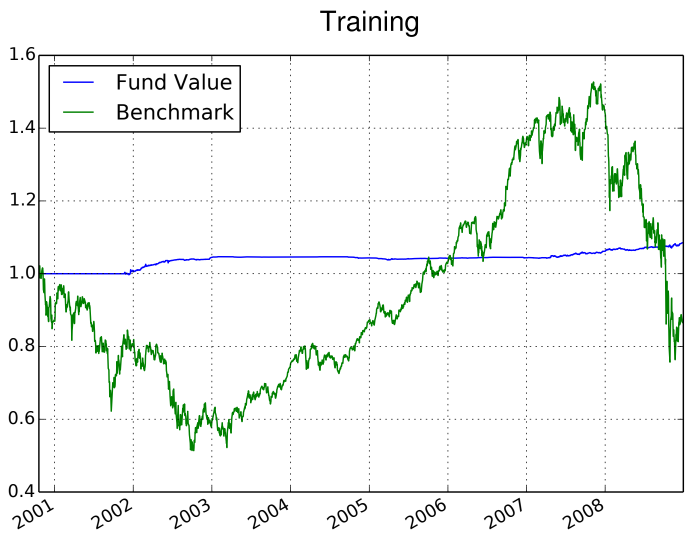
Figure 6.3: Mean shrinkage between training and testing datasets.

6.6.2 *Random Sampling Fitness Genetic Programming*

Our proposed method RSFGP achieves the highest out-of-sample performance in all the metrics as seen in Figures 6.1, 6.2, 6.5 and 6.6 with a mean sterling ratio of 0.6124, a mean total return of 31.81% and a mean Sharpe ratio of 0.6933, substantially beating the IBEX35 benchmark portfolio.

RSFGP also delivers the least shrinkage as can be seen in Figure 6.3 and Table 6.4 with a -28,84 % shrinkage in sterling ratio and -1,29% in total return. We have to note that the performance measured in Sharpe ratio did not only experience shrinkage, but quite the opposite, with an average increase of 13,03% during out-of-sample testing as compared to the training dataset.

UNIVERSIDAD DE MÁLAGA



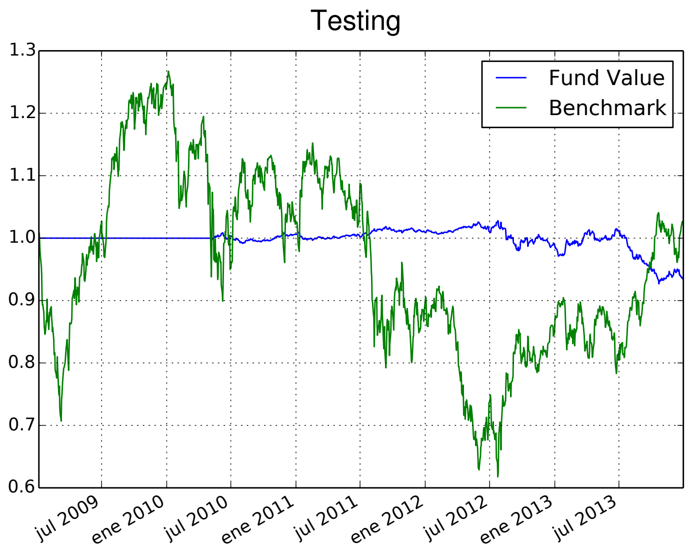


Figure 6.4: Training (top) and out-of-sample testing (bottom) performance of an over-fitted SGP portfolio.

Figure 6.7 shows a robust strategy evolved with RFSGP. The out-of-sample results show higher returns and substantially lower volatility in out-of-sample testing when compared with the IBEX35 benchmark portfolio.

Figure 6.8 compares the mean daily returns and standard deviation of returns (volatility) between all the experiments. We can clearly see that a

UNIVERSIDAD DE MÁLAGA

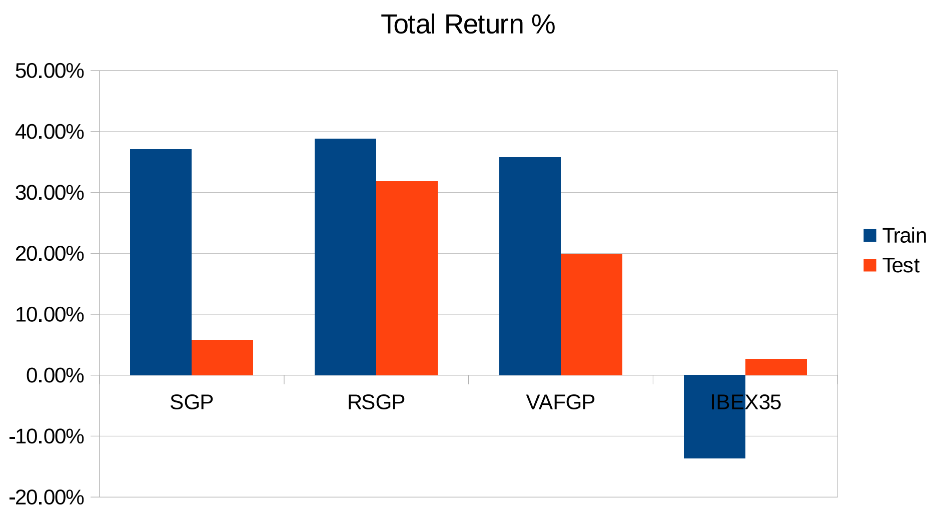


Figure 6.5: Mean total return obtained in all executions.

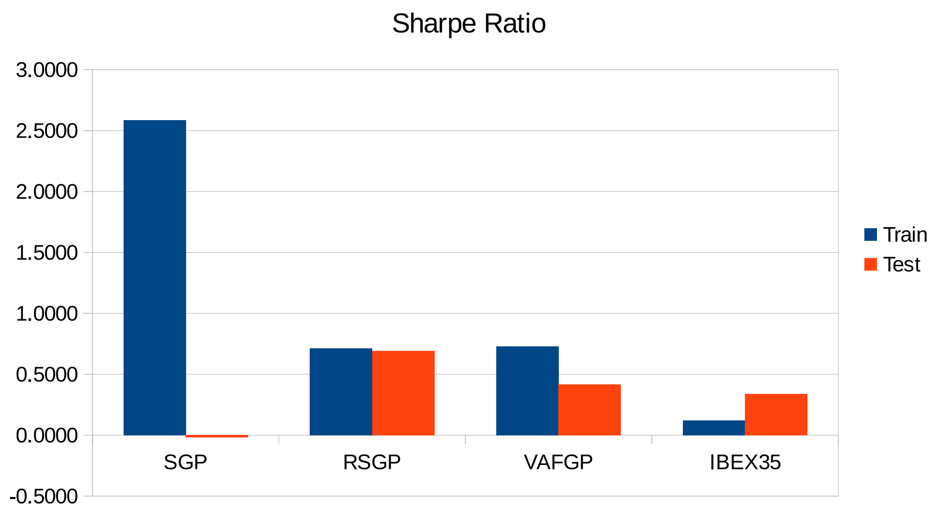


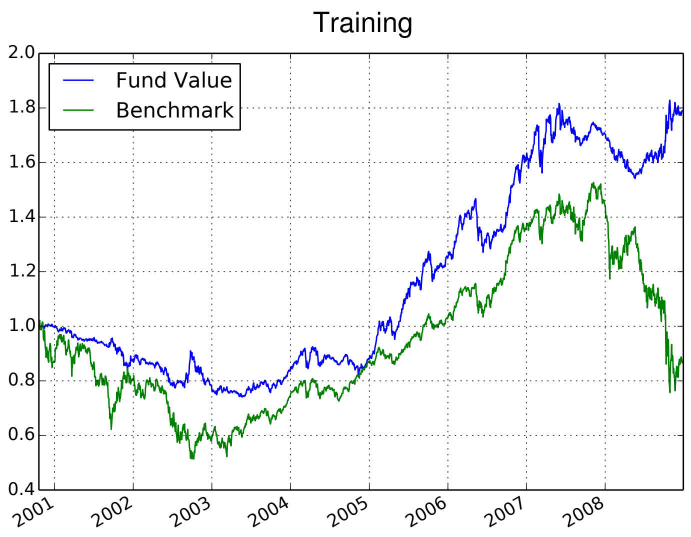
Figure 6.6: Mean sharpe ratio obtained in all executions.

significant part of the solutions generated by RSFGP have a higher return and similar volatility to VAFGP. RSFGP produces portfolios that generate higher returns but at similar risk as VAFGP.

6.6.3 *Volatility Adjusted Fitness Genetic Programming*

VAFGP offered the second best out-of-sample testing results after RSFGP with a mean sterling ratio of 0.3809, only slightly better than SGP which

UNIVERSIDAD DE MÁLAGA



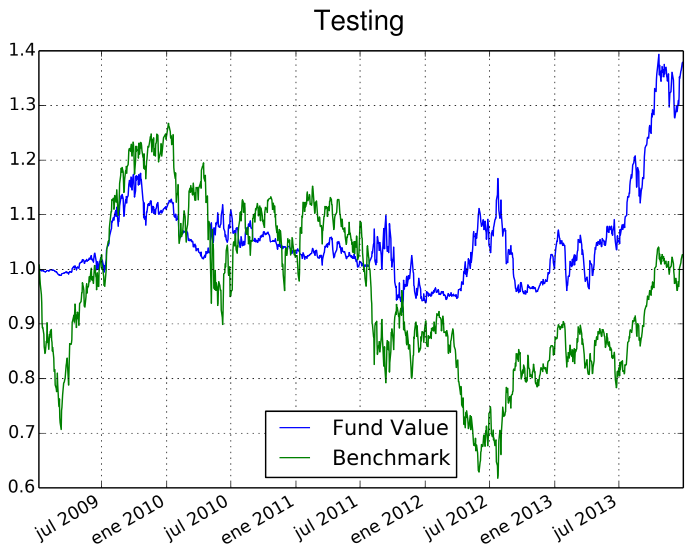


Figure 6.7: Training (top) and out-of-sample testing (bottom) performance of a RSFGP strategy.

had a mean sterling ratio of 0.2723 while the IBEX35 benchmark portfolio had a 0.0521 sterling ratio . In terms of mean total return, VAFGP had a 19.85% return compared with 2.67% for the IBEX35.

The mean daily return and volatility scatter plot in Figure 6.8 shows that VAFGP had a worse mean daily return compared to RSFGP at a similar

UNIVERSIDAD DE MÁLAGA

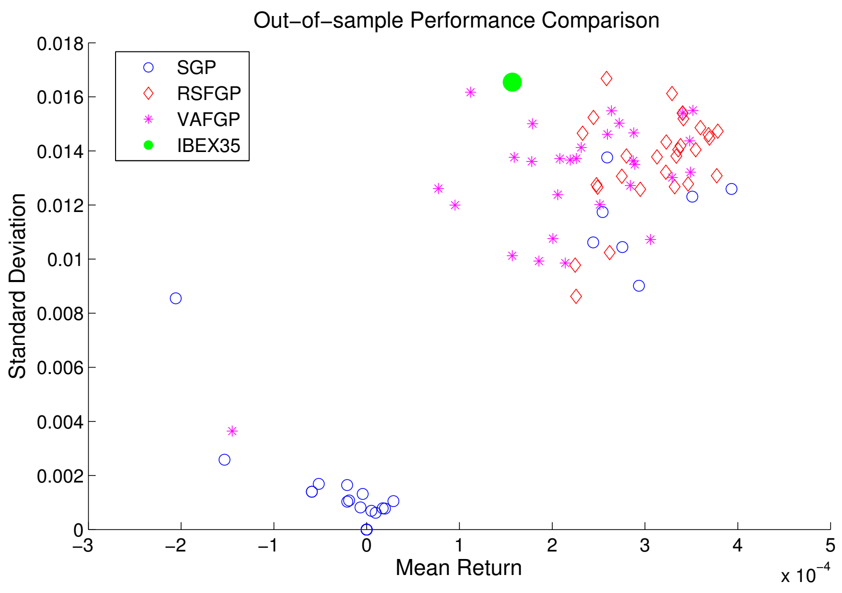


Figure 6.8: Comparison of out-of-sample mean daily portfolio returns and standard deviation (volatility).

level of volatility. One of the VAFGP solutions had a terrible out-of-sample performance as it can be seen in the bottom left quadrant of the graph. VAFGP solutions have a higher dispersion in performance, whereas RSFGP solutions are more concentrated in the same area of the plot.

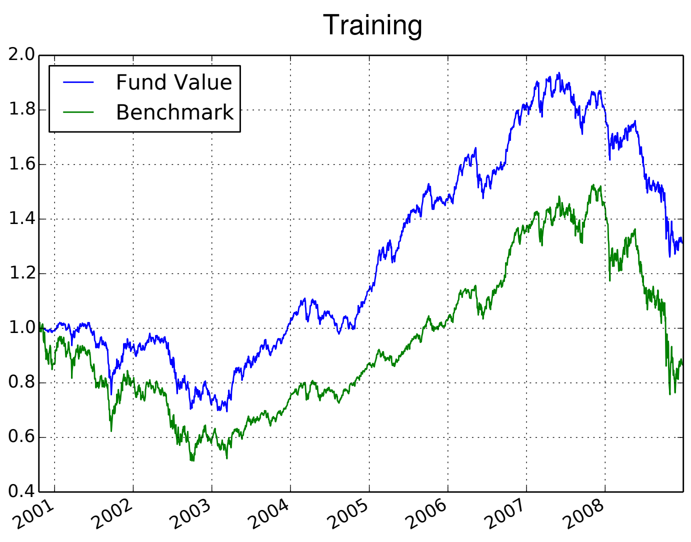
Our results show that including the standard deviation in the fitness (see Equations 6.2 and 6.3) offers reduced performance and higher shrinkage compared to RSFGP. This might be due to the same problem the Sharpe ratio experiences, as introducing a divisor in the fitness distorts it. Figure 6.9 shows the training and testing performance of a VAFGP portfolio.

6.6.4 Statistical significance of results

We analysed the statistical significance of the out-of-sample results by performing a Wilcoxon rank-sum test on the sterling ratios obtained from evaluating the solutions out-of-sample. The Wilcoxon rank-sum test is a non-parametric test of the null hypothesis that two populations are the same (have the same median) against an alternative hypothesis that one population has larger values than the other. This test is very practical as it does not assume a normal-distribution and has greater efficiency than the *t-test* on non-normal distributions.

Table 6.6 shows the *P-values* obtained in the Wilcoxon rank-sum test. All tests have a low P-value thus rejecting the null hypothesis that results between SGP, RSFGP and VAFGP are the same at the 5% significance level.

UNIVERSIDAD DE MÁLAGA



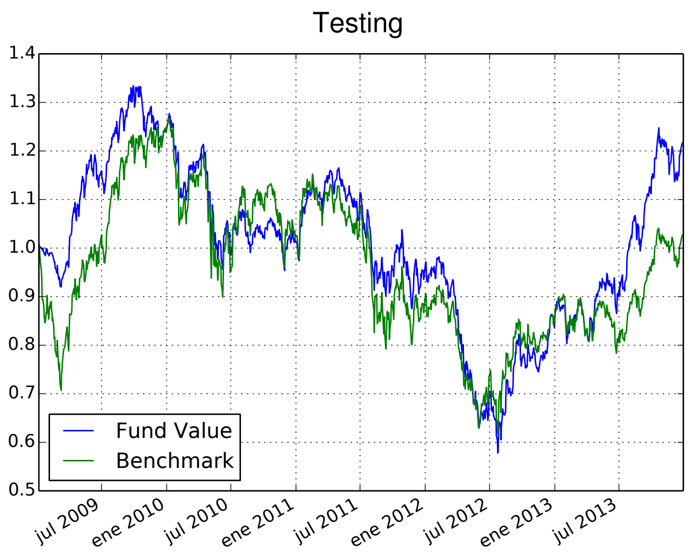


Figure 6.9: Training (top) and out-of-sample testing (bottom) performance of a VAFGP strategy.

The best results are given by RSFGP, followed by VAFGP and lastly SGP as can be seen in Figure 6.1.

UNIVERSIDAD DE MÁLAGA

	SGP	RSFGP	VAFGP
SGP	1,00000	0,00490	0,09730
RSFGP	0,00490	1,00000	0,00047
VAFGP	0,09730	0,00047	1,00000

Table 6.6: Wilcoxon rank-sum test *P-values* for out-of-sample-testing.

6.7 Conclusions

This chapter presents a novel GP method for learning robust *Technical Trading Strategy* using a random sampling method (RSFGP) which improves the performance of solutions when tested out-of-sample and reduces over-fitting. RSFGP calculates the fitness across the randomly sampled segments from the time series and produces solutions that perform in similar fashion during testing and training and that can be used for a prolonged duration across different market conditions. In this study we have also included along some popular technical indicators, common in the literature, some novel financial metrics never used together previously on other GP studies such as the returns, the moving average of returns, the Capital Asset Pricing Method (CAPM) alpha and beta, the Sharpe ratio and the volatility of the stocks calculated over different period lengths.

The proposed approach is able to cope well with extreme drops in the market, reducing the possible loses of capital, and was validated using real and public available market data from 21 of the most liquid stocks from the Spanish stock market. The results show a return of slightly higher than 30% for the testing period of 2009 to 2013, having this period experienced the sovereign debt crisis that affected Spain, which brought the market down to level to the worst days of the 2008 sub-prime crisis with the collapse of Lehman Brothers. In the same period our benchmark, Spanish IBEX35 market index gained less than 3%.

We have also studied the effects of including the standard deviation as a divisor in the fitness as proposed in Yan and Clack (2010) with their *Volatility Adjusted Fitness* (VAFGP). The results of this study show worse performance for VAFGP compared to RSFGP. We demonstrate that for our dataset, including the standard deviation of the fitness as a divisor degrades the performance of the GP evolved solutions.

Both VAFGP and RSFGP were able to produce solutions that on average beat the IBEX35 benchmark portfolio during out-of-sample testing in terms of risk and return. The RSFGP is the method that offered the best results with an mean return of 31.81% compared to 2.67% of the IBEX35 reference benchmark. RSFGP is also the method that experienced the least shrinkage from training to out-of-sample testing, clearly demonstrating that our method increases the robustness of solutions and reduces over-fitting.

UNIVERSIDAD DE MÁLAGA

The RSFGP method had the least shrinkage when compared to SGP and VAFGP

Some key advantages of this method are:

- It discovers robust *Technical Trading Strategy* which can be applied to managing a portfolio of stocks in an automatic manner without requiring the help of financial market experts.
- Solutions are robust, as they can be used for prolonged periods without needing the system to be retrained.
- Solutions are able to cope with extreme market environments.
- Solutions provide similar performance during training and out-of-sample testing.

Some limitations of this research were the consideration of only long positions for the portfolio, as sometimes market regulators place bans on short selling, as was the case for bank stocks during the European sovereign debt crisis, and bank bailouts by the government after the sub-prime crisis. Another research limitation of this work was the number stocks considered, this was due to choosing only the most liquid stocks that had trading data for the years 2000 to 2013, resulting in the 21 stocks of table 6.1.

UNIVERSIDAD DE MÁLAGA

Part III

Conclusions and future work

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD DE MÁLAGA

Chapter 7

Conclusions and future work

Patterns of price movement are not random. However, they're close enough to random...

Jim Simmons (Renaissance Technologies)

7.1 Conclusions

The main conclusion that can be extracted from this thesis is that there is room for robust evolutionary optimization techniques to improve the current approaches in TTS optimization and discovery. It motivates the acceptance that EA techniques combined with robust optimization approaches are able to cope with the uncertainty present in financial data, and to produce TTSs that significantly beat the market with reduced risk.

Overall it can be concluded that:

- Standard GA and GP techniques are guided to peaks of high-fitness producing TTSs that perform very well for the in-sample data but poorly when tested out-of-sample.
- In order to solve over-fitting, a robustness mechanism is needed to guide the algorithm towards areas of the search space where the fitness is higher on average.
- The problem can be tackled as a single-objective problem where the original objective function is averaged by sampling the original dataset at the individual level. Each individual is evaluated on different ramdom samples from the original dataset.
- Robustness may also be incorporated by tackling the problem as a multi-objective problem. The original objective function(s) are replaced by the mean and the standard deviation (calculated from a

UNIVERSIDAD DE MÁLAGA

basket of assets and/or sampling the original dataset) of the original objective function.

- Robustness can also be incorporated by optimizing together a basket of indices or stocks. Instead of generating a different TTS for each one, producing a single TTS that is able to operate of an all of them leads to soltuions with increased robustness.
- Robustness can be measured by calculating *shrinkage* (the percent reduction in performance between testing and training).
- Robust solutions perform satisfactorily when tested out-of-sample. They offer very little shrinkage implying the performance is very similar to the performance during the training phase.
- The robustness mechanisms exposed lead to robust TTS that can be used for prolonged periods without requiring the system to be re-optimized.
- The Robust TTS generated are able to cope with extreme market environments.
- Including $\mathcal{MDD}$ as an additional objective function in a multiobjective problem or by as the divisor in the sterling ratio in a single-optimization problem produces solutions with lower risk profile that just optimizing the Sharpe Ratio or Sortino Ratio

7.2 Future work

Due to the wide range of topics covered in this thesis, future research could be done on several fronts:

- On the algorithmic side, conducting a thorough analysis of different enhanced solvers for addressing this problem could be of great interest.
- It is well known that Pareto optimality performs badly when the number of objectives is greater than 3. This is the problem addressed in Chapter 5, so different approaches such as MOEA/D or IBEA could be considered.
- This work could be extended to other markets and datasets in order to continue studying the beneficial effects of the proposed multi-market and random sampling methods.
- The scope of the problem can be extended where a long/short portfolio is simulated and the buy and sell rules are co-evolved.

UNIVERSIDAD DE MÁLAGA

- Future research could include a greater variety of technical, and quantitative analysis derived metrics such as auto-correlation of returns, and test different risk metrics as VaR (Value at Risk) and CVaR (Conditional Value at Risk) instead of maximum draw-down or volatility.

- Financial markets are subject to changing conditions over time. Being able to optimize in a dynamic environment could be very interesting, as changes may affect the problem instance, the objective functions, and/or the constrains. Dynamic optimization techniques could be used to track the optimal changing trading strategies over time.

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA

Bibliography

Aguilar-Rivera, R., Valenzuela-Rendón, M. and Rodríguez-Ortiz, J. Genetic algorithms and Darwinian approaches in financial applications: A survey. *Expert Systems with Applications*, volume 42(21), pages 7684–7697, 2015. ISSN 09574174.

Aguirre, H. Advances on Many-objective Evolutionary Optimization. In *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '13 Companion, pages 641–666. ACM, New York, NY, USA, 2013. ISBN 978-1-4503-1964-5.

Allen, F. and Karjalainen, R. Using genetic algorithms to find technical trading rules. *Journal of Financial Economics*, volume 51(2), pages 245–271, 1999.

Appel, G. *Technical Analysis: Power Tools for Active Investors*. Financial Times Prentice Hall books. Financial Times/Prentice Hall, 2005. ISBN 9780131479029.

Bauer, R. and Liepins, G. *Genetic Algorithms and Computerized Trading Strategies*. Working paper series (University of Western Ontario. School of Business Administration. Research and Publications Division). Research and Publications, School of Business Administration, University of Western Ontario, 1988. ISBN 9780771410581.

Bauer, R. J. *Genetic Algorithms and Investment Strategies*. John Wiley & Sons, Inc., New York, NY, USA, 1994. ISBN 0471576794.

Becker, L. and Seshadri, M. GP-evolved technical trading rules can outperform buy and hold. *Proceedings of the Sixth International Conference on Computational Intelligence and Natural Computing, Embassy Suites Hotel and Conference Center, Cary, North Carolina USA, September 26-30 2003*, 2003.

Berutich, J. M., López, F., Luna, F. and Quintana, D. Robust technical trading strategies using GP for algorithmic portfolio selection. *Expert Systems with Applications*, volume 46, pages 307–315, 2016.

UNIVERSIDAD DE MÁLAGA

BERUTICH, J. M., LUNA, F. and LÓPEZ, F. On the quest for robust technical trading strategies using multi-objective optimization. *AI Communications*, volume 27(4), pages 453–471, 2014.

BEUME, N., NAUJOKS, B. and EMMERICH, M. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, volume 181(3), pages 1653 – 1669, 2007. ISSN 0377-2217.

BEYER, H. and SENDHOFF, B. Robust optimization - A comprehensive survey. *Computer Methods in Applied Mechanics and Engineering*, volume 196(33-34), pages 3190–3218, 2007. ISSN 00457825.

BIANCHI, L., DORIGO, M., GAMBARDELLA, L. M. and GUTJAHR, W. J. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, volume 8(2), pages 239–287, 2009. ISSN 15677818.

BLUM, C. and ROLI, A. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Comput. Surv.*, volume 35(3), pages 268–308, 2003. ISSN 0360-0300.

BOLLINGER, J. *Bollinger on Bollinger Bands*. McGraw-Hill Education, 2001. ISBN 9780071373685.

BOWERS, C. Simulating evolution with a computational model of embryogeny. *Doctral Thesis, The University of Birmingham*, (November), 2006.

BRABAZON, A., O'NEILL, M. and MARINGER, D. *Natural computing in computational finance*. 2012. ISBN 978-3-642-23336-4.

BRANKE, J. Creating Robust Solutions by Means of Evolutionary Algorithms. In *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, PPSN V, pages 119–128. Springer-Verlag, London, UK, UK, 1998. ISBN 3-540-65078-4.

BRANKE, J. *Efficient Evolutionary Algorithms for Searching Robust Solutions*, pages 275–285. Springer London, London, 2000. ISBN 978-1-4471-0519-0.

BRIZA, A. C. and NAVAL, P. C., JR. Stock trading system based on the multi-objective particle swarm optimization of technical indicators on end-of-day market data. *Applied Soft Computing*, volume 11(1), pages 1191–1201, 2011. ISSN 1568-4946.

BROCK, W., LAKONISHOK, J. and LEBARON, B. Simple Technical Trading Rules and the Stochastic Properties of Stock Returns. *Journal of Finance*, volume 47(5), pages 1731–64, 1992.

CHEN, S. *Genetic Algorithms and Genetic Programming in Computational Finance*. Kluwer Academic Publishers, 2002. ISBN 9780792376019.

UNIVERSIDAD DE MÁLAGA

CHEN, S.-H., WANG, P. P. and KUO, T.-W. *Computational Intelligence in Economics and Finance: Volume II*, volume 2. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. ISBN 978-3-540-72821-4.

CHEN, W. and SUNDARARAJ, G. J. Physical Programming for Robust Design. In *40th Structures, Structural Dynamics and Materials Conference, St. Louis, USA*, volume 1206, pages 17–26. St. Louis, USA, 1999.

CHIAM, S. C., TAN, K. C. and AL MAMUN, A. Investigating technical trading strategy via an multi-objective evolutionary platform. *Expert System with Applications*, volume 36(7), pages 10408–10423, 2009. ISSN 0957-4174.

COELLO, C., LAMONT, G. and VAN VELDHUIZEN, D. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Genetic and Evolutionary Computation. Springer US, 2007. ISBN 9780387367972.

CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L. and STEIN, C. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009. ISBN 0262033844, 9780262033848.

DEB, K. and GUPTA, H. *Searching for Robust Pareto-Optimal Solutions in Multi-objective Optimization*, pages 150–164. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. ISBN 978-3-540-31880-4.

DEB, K. and GUPTA, H. Introducing robustness in multi-objective optimization. *Evolutionary computation*, volume 14(4), pages 463–94, 2006. ISSN 1063-6560.

DEB, K., PRATAP, A., AGARWAL, S. and MEYARIVAN, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on*, volume 6(2), pages 182–197, 2002. ISSN 1089-778X.

DEMPSTER, M. and JONES, C. A real-time adaptive trading system using genetic programming. *Quantitative Finance*, volume 1(4), pages 397–413, 2001. ISSN 1469-7688.

FORTIN, F. A., DE RAINVILLE, F. M., GARDNER, M. A., PARIZEAU, M. and GAGNÉ, C. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, volume 13, pages 2171–2175, 2012.

GAREY, M. R. and JOHNSON, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979. ISBN 0716710447.

GASPAR-CUNHA, A. and COVAS, J. A. Robustness in multi-objective optimization using evolutionary algorithms. *Computational Optimization and Applications*, volume 39(1), pages 75–96, 2008. ISSN 09266003.

UNIVERSIDAD DE MÁLAGA

GENCAY, R. The predictability of security returns with simple technical trading rules. *Journal of Empirical Finance*, volume 5(4), pages 347–359, 1998.

GENDREAU, M. and POTVIN, J.-Y. Handbook of Metaheuristics. *Handbook of Metaheuristics*, volume 146, page 648, 2010. ISSN 978-1-4419-1665-5.

GLOVER, F. Future Paths for Integer Programming and Links to Artificial Intelligence. *Comput. Oper. Res.*, volume 13(5), pages 533–549, 1986. ISSN 0305-0548.

GLOVER, F. and KOCHENBERGER, G. A. *Handbook of Metaheuristics*, volume 57. Springer US, 2003. ISBN 978-0-306-48056-0.

GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989. ISBN 0201157675.

GRANGER, C. W. and HYUNG, N. Occasional structural breaks and long memory with an application to the S&P 500 absolute stock returns. *Journal of Empirical Finance*, volume 11(3), pages 399–421, 2004. ISSN 09275398.

GYPTEAU, J., OTERO, F. and KAMPOURIDIS, M. Generating Directional Change Based Trading Strategies with Genetic Programming. In *Applications of Evolutionary Computation* (edited by A. M. Mora and G. Squillero), volume 9028 of *Lecture Notes in Computer Science*, pages 267–278. Springer International Publishing, 2015. ISBN 978-3-319-16548-6.

HAUPT, R. and HAUPT, S. *Practical Genetic Algorithms*. Wiley InterScience electronic collection. Wiley, 2004. ISBN 9780471671756.

HENDERSHOTT, T. and RIORDAN, R. Algorithmic Trading and the Market for Liquidity. *Journal of Financial and Quantitative Analysis*, volume 48(4), pages 1001–1024, 2013.

HOLLAND, J. H. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, USA, 1975.

HSU, C.-M. A hybrid procedure for stock price prediction by integrating self-organizing map and genetic programming. *Expert Systems with Applications*, volume 38(11), pages 14026 – 14036, 2011. ISSN 0957-4174.

HU, Y., LIU, K., ZHANG, X., SU, L., NGAI, E. and LIU, M. Application of evolutionary computation for rule discovery in stock algorithmic trading: A literature review. *Applied Soft Computing*, volume 36, pages 534–551, 2015. ISSN 15684946.

UNIVERSIDAD DE MÁLAGA

Iba, H. and Aranha, C. C. *Practical Applications of Evolutionary Computation to Financial Engineering: Robust Techniques for Forecasting, Trading and Hedging*. Springer Publishing Company, Incorporated, 2012. ISBN 3642276474, 9783642276477.

Jaimes, A. L. and Coello, C. A. C. *An Introduction to Multi-Objective Evolutionary Algorithms and Some of Their Potential Uses in Biology*, pages 79–102. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-78534-7.

Jin, Y. and Sendhoff, B. *Trade-Off between Performance and Robustness: An Evolutionary Multiobjective Approach*, pages 237–251. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. ISBN 978-3-540-36970-7.

Kaufman, P. *New Trading Systems and Methods*. Wiley Trading. Wiley, 2005. ISBN 9780471268475.

Khouadjia, M. R., Sarasola, B., Alba, E., Talbi, E.-G. and Jourdan, L. *Metaheuristics for Dynamic Vehicle Routing*, pages 265–289. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-30665-5.

Kitano, H. Biological robustness. *Nature reviews. Genetics*, volume 5(11), pages 826–37, 2004. ISSN 1471-0056.

Koza, J. R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992. ISBN 0-262-11170-5.

Kushchu, I. Genetic programming and evolutionary generalization. *IEEE Transactions on Evolutionary Computation*, volume 6(5), pages 431–442, 2002. ISSN 1089-778X.

Lane, G. M. Lane's Stochastics. *Technical Analysis of Stocks and Commodities*, (2), pages 87–90, 1984.

Li, J. and Taiwo, S. Enhancing Financial Decision Making Using Multi-Objective Financial Genetic Programming. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 2171 –2178. 2006.

Li, M., Yang, S., Liu, X., Shen, e. R. C., Ruimin", Fleming, P. J., Fonseca, C. M., Greco, S. and Shaw, J. *A Comparative Study on Evolutionary Algorithms for Many-Objective Optimization*, pages 261–275. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-37140-0.

Lintner, J. The valuation of risk assets and the selection of risky investments in stock portfolios and capital budgets. *Review of Economics and Statistics*, volume 47(1), pages 13–37, 1965.

UNIVERSIDAD DE MÁLAGA

LO, A. The Statistics of Sharpe Ratios. In *Social Science Research Network Working Paper Series*. 2003.

LO, A. W., MAMAYSKY, H. and WANG, J. Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation. *Journal of Finance*, volume 40, pages 1705–1765, 2000.

LOHPETCH, D. and CORNE, D. Discovering effective technical trading rules with genetic programming: towards robustly outperforming buy-and-hold. *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, pages 439–444, 2009.

LONES, M. A. Metaheuristics in nature-inspired algorithms. *Proceedings of the 2014 conference companion on Genetic and evolutionary computation companion - GECCO Comp '14*, pages 1419–1422, 2014.

LORENZO, R. Rsid (rsi detrended). In *How to Make Money by Fast Trading*, Perspectives in Business Culture, pages 197–200. Springer Milan, 2012. ISBN 978-88-470-2533-2.

LUENGO, S., WINKLER, S., BARRERO, D. and CASTAÑO, B. Optimization of trading rules for the spanish stock market by genetic programming. In *Current Approaches in Applied Artificial Intelligence* (edited by M. Ali, Y. S. Kwon, C.-H. Lee, J. Kim and Y. Kim), volume 9101 of *Lecture Notes in Computer Science*, pages 623–634. Springer International Publishing, 2015. ISBN 978-3-319-19065-5.

MALLICK, D. and LEE, V. C. An empirical study of Genetic Programming generated trading rules in computerized stock trading service system. *2008 International Conference on Service Systems and Service Management*, pages 1–6, 2008.

MASHWANI, W. K., SALHI, A., ASIF JAN, M., SULAIMAN, M., ADEEB KHANUM, R. and ALGARNI, A. Evolutionary Algorithms Based on Decomposition and Indicator Functions: State-of-the-art Survey. *International Journal of Advanced Computer Science and Applications*, volume 7(2), 2016.

MATSUI, K. and SATO, H. Neighborhood evaluation in acquiring stock trading strategy using genetic algorithms. In *Soft Computing and Pattern Recognition (SoCPaR), 2010 International Conference of*, pages 369 –372. 2010.

MAULIK, U., BANDYOPADHYAY, S. and MUKHOPADHYAY, A. *Multiobjective Genetic Algorithms for Clustering: Applications in Data Mining and Bioinformatics*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-16615-0.

UNIVERSIDAD DE MÁLAGA

Mehta, K. and Bhattacharyya, S. Adequacy of training data for evolutionary mining of trading rules. *Decission Support Systems*, volume 37(4), pages 461–474, 2004. ISSN 0167-9236.

Mossin, J. Equilibrium in a Capital Asset Market. *Econometrica*, volume 34(4), pages 768–783, 1966.

Mousavi, S., Esfahanipour, A. and Zarandi, M. H. F. A novel approach to dynamic portfolio trading system using multitree genetic programming. *Knowledge-Based Systems*, volume 66, pages 68–81, 2014. ISSN 09507051.

Murphy, J. *Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications*. New York Institute of Finance Series. New York Institute of Finance, 1999. ISBN 9780735200661.

Mutingi, M. and Mbohwa, C. *Grouping Genetic Algorithms: Advances and Applications*, volume 666 of *Studies in Computational Intelligence*. Springer International Publishing, 2017. ISBN 978-3-319-44394-2.

Neely, C. J. Risk-adjusted, ex ante, optimal technical trading rules in equity markets. *International Review of Economics & Finance*, volume 12(1), pages 69–87, 2003. ISSN 10590560.

Ni, J., Cao, L. and Zhang, C. Evolutionary Optimization of Trading Strategies. In *Proceedings of the 2008 conference on Applications of Data Mining in E-Business and Finance*, pages 11–24. IOS Press, Amsterdam, The Netherlands, The Netherlands, 2008. ISBN 978-1-58603-890-8.

Nuti, G., Mirghaemi, M., Treleaven, P. and Yingsaeree, C. Algorithmic Trading. *IEEE Computer Society*, (November), pages 61–69, 2011.

O'Reilly, U.-M., Yu, T., Riolo, R. and Worzel, B. *Genetic Programming Theory and Practice II*. Springer US, 2005. ISBN 0792381351.

Pardo, R. *The Evaluation and Optimization of Trading Strategies*. Wiley Trading. Wiley, 2008. ISBN 9781118045053.

Pictet, O. V., Dacorogna, M. M., Dave, R. D., Chopard, B., Schirru, R. and Tomassini, M. Genetic Algorithms with collective sharing for Robust Optimization in Financial Applications. Working Papers 1995-02-06., Olsen and Associates, 1995.

Pinto, J. M., Neves, R. F. and Horta, N. Boosting Trading Strategies performance using VIX indicator together with a dual-objective Evolutionary Computation optimizer. *Expert Systems with Applications*, volume 42(19), pages 6699–6716, 2015. ISSN 09574174.

UNIVERSIDAD DE MÁLAGA

Piszcz, A. and Soule, T. Dynamics of evolutionary robustness. In *GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation* (edited by M. Keijzer, M. Cattolico, D. Arnold, V. Babovic, C. Blum, P. Bosman, M. V. Butz, C. Coello Coello, D. Dasgupta, S. G. Ficici, J. Foster, A. Hernandez-Aguirre, G. Hornby, H. Lipson, P. McMinn, J. Moore, G. Raidl, F. Rothlauf, C. Ryan and D. Thierens), volume 1, pages 871–878. ACM Press, Seattle, Washington, USA, 2006. ISBN 1-59593-186-4.

Poli, R. and Langdon, W. Genetic programming: An introductory tutorial and a survey of techniques and applications. *University of Essex, UK*, pages 1–112, 2007.

Ponsich, A., Jaimes, A. L. and Coello, C. A. C. A Survey on Multiobjective Evolutionary Algorithms for the Solution of the Portfolio Optimization Problem and Other Finance and Economics Applications. *IEEE Transactions on Evolutionary Computation*, volume 17(3), pages 321–344, 2013. ISSN 1089-778X.

Riolo, R., McConaghy, T. and Vladislavleva, E. *Genetic Programming Theory and Practice VIII*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010. ISBN 1441977465, 9781441977465.

Riolo, R., Worzel, B., Kotanchek, M. and Kordon, A. *Genetic Programming Theory and Practice XIII*. Springer International Publishing, 1st edition, 2016. ISBN 978-3-319-34223-8, 978-3-319-34221-4.

Rothlauf, F. *Optimization Methods*, pages 45–102. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-540-72962-4.

Sharpe, W. F. Capital asset prices: A theory of market equilibrium under conditions of risk. *Journal of Finance*, volume 19(3), pages 425–442, 1964.

Sharpe, W. F. Mutual Fund Performance. *Journal of Business*, volume 39(S1), pages 119–138, 1966.

Sharpe, W. F. The Sharpe Ratio. *The Journal of Portfolio Management*, volume 21(1), pages 49–58, 1994.

Sörensen, K. and Glover, F. *Encyclopedia of Operations Research and Management Science*. Springer-Verlag, Berlin, Heidelberg, 3rd edition, 2013. ISBN 1441911375, 9781441911377.

Sortino, F. A. and Price, L. N. Performance Measurement in a Downside Risk Framework. *The Journal of Investing*, volume 3(3), pages 59–64, 1994.

UNIVERSIDAD DE MÁLAGA

Soule, T. Operator choice and the evolution of robust solutions. *Genetic Programming Theory and Practice*, pages 257–269, 2003.

Subramanian, H., Ramamoorthy, S., Stone, P. and Kuipers, B. J. Designing safe, profitable automated stock trading agents using evolutionary algorithms. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, GECCO '06, pages 1777–1784. ACM, New York, NY, USA, 2006. ISBN 1-59593-186-4.

Thomsett, M. *Mastering Fundamental Analysis*. Kaplan Financial Series. Dearborn Finanical Pub., 1998. ISBN 9780793128730.

Tsinaslanidis, P. E. and Zapranis, A. D. *Technical Analysis for Algorithmic Pattern Recognition*. Springer International Publishing, Cham, 2016. ISBN 978-3-319-23636-0.

Tsutsui, S. and Ghosh, A. Genetic algorithms with a robust solution searching scheme. *Trans. Evol. Comp*, volume 1(3), pages 201–208, 1997. ISSN 1089-778X.

Wikipedia. Metaheuristic — Wikipedia, The Free Encyclopedia. 2016. Online; accessed 30-September-2016.

Wilder, J. *New Concepts in Technical Trading Systems*. Trend Research, 1978. ISBN 9780894590276.

Yan, W. and Clack, C. D. Evolving robust GP solutions for hedge fund stock selection in emerging markets. *Soft Computing*, volume 15(1), pages 37–50, 2010. ISSN 1432-7643.

Zhang, H. and Ren, R. High Frequency Foreign Exchange Trading Strategies Based on Genetic Algorithms. *2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing*, pages 426–429, 2010.

Zhang, Q. and Li, H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, volume 11(6), pages 712–731, 2007.

Zitzler, E. and Künzli, S. *Indicator-Based Selection in Multiobjective Search*, pages 832–842. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-540-30217-9.

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA

List of Acronyms

ACO	*Ant Colony Optimization*
AT	*Algorithmic Trading*
BB	*Bollinger Bands ®*
BH	*Buy & Hold*
CAPM	*Capital Asset Pricing Model*
CFD	*Contract For Differences*
EA	*Evolutionary Algorithms*
EC	*Evolutionary Computation*
EMA	*Exponential Moving Average*
EP	*Evolutionary Programming*
ES	*Evolution Strategy*
ETF	*Exchange Traded Fund*
FX	*Foreign Exchange*
GA	*Genetic Algorithm*
GP	*Genetic Programming*
HH	*Highest High*
IBS	*Internal Bar Strength*
LL	*Lowest Low*
MACD	*Moving Average Convergence Divergence*
$\mathcal{MDD}$	*Maximum Drawdown*
MOEA	*Multiobjective Evolutionary Algorithm*

UNIVERSIDAD DE MÁLAGA

MOP	*Multiobjective Optimization Problem*
OHLCV	*Open, High, Low, Close, Volume*
PSO	*Particle Swarm Optimization*
QA	*Quantitative Analysis*
RSFGP	*Random Sampling Fitness Genetic Programming*
RSI	*Relative Strength Index*
SA	*Simulated Annealing*
SGP	*Standard Genetic Programming*
SMA	*Simple Moving Average*
SO	*Stochastic Oscillator*
$\mathcal{SR}$	*Sortino Ratio*
TA	*Technical Analysis*
TTS	*Technical Trading Strategy*
VAFGP	*Volatility Adjusted Fitness Genetic Programming*

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA