Universidad de La Laguna

Facultad de Ciencias Económicas y Empresariales

MEMORIA DEL TRABAJO FIN DE GRADO

Trading Algorítmico: Investigación Práctica y teorema No Free Lunch

(Algorithmic Trading: Practice Research and No Free Lunch Theorem)

Autor: D. Alberto Ruiz Benítez de Lugo Hernández

Tutor: D. Javier Giner Rubio

Grado en ECONOMÍA

FACULTAD DE ECONOMÍA, EMPRESA Y TURISMO

Curso Académico 2014 / 2015

San Cristóbal de La Laguna, a 30 de Junio de 2015

D. Javier Giner Rubio del Departamento de Economía Financiera y Contabilidad

CERTIFICA:

Que la presente Memoria de Trabajo Fin de Grado en Economía titulada *"Algorithmic Trading: Practice Research and No Free Lunch Theorem"* y presentada por el alumno Alberto Ruiz Benítez de Lugo Hernández

realizada bajo mi dirección, reúne las condiciones exigidas por la Guía Académica de la asignatura para su defensa

Para que así conste y surta los efectos oportunos, firmo la presente en La Laguna a treinta de Junio de dos mil quince

El tutor

Fdo: D. Javier Giner Rubio

San Cristóbal de La Laguna, a 30 de Junio de 2015

RESUMEN

En la actualidad las finanzas cuantitativas y las nuevas tecnologías van de la mano. La búsqueda de nuevos métodos en el sector financiero ha propulsado el uso de las ciencias computacionales en el ámbito de las finanzas.

La naturaleza de este trabajo es estudiar cómo funcionan los sistemas de trading algorítmicos. Para ello vamos a desarrollar, mediante programación, cuatro sistemas de trading para comprender qué son y cómo funcionan. Una vez desarrollados, analizaremos en profundidad sus diferencias y particularidades, todo ello a través del punto de vista del teorema No Free Lunch. Finalmente, analizaremos cuáles son las consecuencias de combinar más de un sistema al mismo tiempo.

El estudio de los sistemas de trading algorítmicos es un tema que actualmente se encuentra en la vanguardia de la investigación, pues unifica tres nexos muy importantes: finanzas, matemáticas e informática. Lo que hace muy atractiva a esta área de los mercados financieros.

PALABRAS CLAVE

Trading Algorítmico – Programación – No Free Lunch Theorem – Mercados financieros

ABSTRACT

Nowadays quantitative finance and new technologies go hand in hand. The search for new methods in financial sector has boosted the computer science in the field of finance.

The purpose of this work is to study how algorithmic trading systems works. Therefore, we're going to develop four trading systems, through programming, to understand what they are and how they really work. Once developed, we discussed deeply the differences between each other through the viewpoint of No Free Lunch Theorem. Finally, we looked into the results about mix more than one system up at the same time.

This area of knowledge is into the Forefront of research, linking up three important matters nowadays: finance, math and computing science. Which set this field very attractive to analyze and research in financial markets.

KEY WORDS

Algorithmic Trading – Programming – No Free Lunch Theorem – Financial Markets

ÍNDICE

ÍNDICE DE TABLAS E ILUSTRACIONES

TABLAS

ILUSTRACIONES

1. INTRODUCCIÓN

1.1. NATURALEZA DEL PROYECTO

Las nuevas tecnologías han fomentado numerosos cambios a lo largo del siglo XXI, muchos de ellos procedentes de las telecomunicaciones y de la informática. Estos avances han provocado un cambio en nuestra forma de trabajar, de vivir e incluso de pensar.

La naturaleza de este proyecto es realizar un acercamiento hacia estos avances en el ámbito de las finanzas. En concreto, los cambios producidos en los mercados financieros, donde la introducción de sistemas automáticos de trading ha reinventado la forma de analizar los mercados de valores.

Estos nuevos instrumentos permiten ejecutar órdenes de mercado mucho más rápido que los operadores tradicionales, lo que genera notables cambios a la hora de gestionar los volúmenes de negociación, las variaciones de precios, y los márgenes de maniobra definidos mediante la oferta y la demanda (Bernstein, 2008).

1.2. OBJETIVOS Y DESARROLLO

El objetivo principal de este proyecto es realizar un estudio práctico sobre el diseño, desarrollo y ejecución de un sistema algorítmico de trading.

Para ello, en el apartado dos de este proyecto definiremos los componentes, estructura y funcionalidad de los algoritmos en condiciones ideales de mercado[1], sin tener en cuenta los efectos de deslizamiento[2]. Estas condiciones facilitarán que todas las decisiones de compra-venta que tome nuestro algoritmo se ejecuten con una bondad de ajuste[3] perfecta (Cáceres , 2007). Por consiguiente, en el apartado tres definiremos el procedimiento de programación de estas estrategias y los componentes principales del proceso.

Por último, en el apartado cuatro desarrollaremos varios casos prácticos llevados a cabo sobre la acción Netflix (NFLX) del índice NASDAQ norteamericano. Señalar que para el cálculo de los resultados realizaremos un análisis *backtesting*[4] de cada sistema, lo que permitirá, además de obtener sus resultados contables, analizar las diferencias entre los diferentes algoritmos.

En total, hemos desarrollado cuatro sistemas de trading, cada uno con unas características diferenciadas. Nuestro fin último es comparar dichos sistemas y definir cual es ellos es el más adecuado, enmarcando los resultados desde el punto de vista del *No Free Lunch Theorem*, el cual establece que no existe un modelo perfecto para todas las situaciones de mercado, sino que los modelos deben adecuarse a las circunstancias concretas de cada activo.

---

[1] Situación hipotética de que no existan intermediarios entre el sistema y el mercado real, es decir, sin comisiones añadidas al coste de operación y con un tipo de interés del 2% para las posiciones en liquidez.

[2] Efecto deslizamiento (Slippage): Es el efecto producido por cambios bruscos en el volumen de negociación, lo que provoca que el precio al que se decide comprar o vender un contrato no coincida con el precio cruzado. Es decir, el precio de compra efectivo.

[3] Bondad de ajuste: Estadístico que representa la calidad predictiva de un modelo, en nuestro caso, que la orden de operación sea igual al resultado de operación.

[4] Proceso mediante el cual se analiza qué resultados habría obtenido un sistema si se hubiera aplicado en el pasado.

2. SISTEMAS AUTOMÁTICOS DE TRADING

Un sistema automático de trading es un programa, escrito mediante un código, que sigue determinadas reglas sintácticas que permite analizar y operar en una serie de valores bursátiles en tiempo real. A lo largo de este trabajo los definiremos como sistemas, algoritmos o autómatas. En dicho programa, podemos modelar la realidad y hacer que el sistema tome decisiones por sí mismo en base a unas normas que le hemos impuesto.

Para ello, debemos introducir los inputs del modelo, definir sus variables y establecer unas funciones y reglas de actuación. Todo esto se tratará en detalle en el capítulo tres del presente proyecto.

Como bien señala Jones (1999), un sistema automático de trading seguirá todas las instrucciones que le asignemos, de forma que si lo programamos adecuadamente podremos hacer que el sistema haga prácticamente cualquier tarea con los datos que le suministremos. Como por ejemplo, crear o comparar indicadores[5], realizar cálculos en tiempo real, observar tendencias, entre otros.

Sin embargo, su principal ventaja es su velocidad. Un sistema automático bien implementado puede ser un medio perfecto para adelantarse a las tendencias del mercado y obtener beneficios por ser el primero en llegar.

Para comprender en profundidad cómo funcionan estos algoritmos y sus aplicaciones en la actualidad debemos introducir algunos conceptos básicos utilizados a diario por los desarrolladores de sistemas de trading.

2.1. CONCEPTOS

2.1.1. Backtesting

Este término hace referencia al proceso de comprobación de resultados de una estrategia. Según Holland (2013), se basa en calcular, mediante una base de datos históricos, los resultados que se hubieran obtenido si se hubiera puesto en marcha el sistema (algoritmo).

Es decir, el backtesting permite saber cómo nos hubiera ido si hubiésemos utilizado el sistema en el pasado. La plataforma de trading que hemos utilizado para realizar este proyecto posee una base de datos históricos integrada que permite este análisis de resultados.

2.1.2. Frecuencia

Nuestros sistemas trabajan con datos financieros reales, lo que equivale a realizar cálculos sobre una serie temporal que está en constante cambio. Por tanto, la frecuencia a la que queramos analizar dicha serie (semanas, días, horas, minutos) es importante.

Hay que tener en cuenta que todas nuestras estrategias, indicadores, objetivos e incluso los resultados de la inversión, cambiarán según la frecuencia seleccionada. Se trata de marcar el tempo al que irá nuestra música.

---

[5] Podemos crear nuestros propios indicadores bursátiles, pues el software tiene potentes herramientas tanto matemáticas como estadísticas.

2.1.3. StopLoss

Para limitar las posibles pérdidas que se puedan generar, los algoritmos poseen funciones que les habilitan para cerrar posiciones cuando las pérdidas llegan a unos determinados límites. Estas funciones se denominan "StopLoss" y pueden ser determinadas por nosotros mismos. Del mismo modo, pueden ser aplicadas o no, pero resulta extremadamente recomendable incorporarlas.

2.1.4. Tipos de operaciones

2.1.4.1. Entrar al Mercado

El momento en el que adquirimos (compramos) un contrato significa que hemos entrado al mercado. Esta denominación nace de que ahora somos parte del mercado, poseemos un activo que va cambiando de valor y que podemos vender en cualquier momento. Como señala Catalayud (2007), existen dos formas de entrar en el mercado:

- Operar en largo: Son operaciones que realizamos cuando adquirimos un contrato con la intención de venderlo en un futuro próximo.
- Operar en corto*:* Son operaciones que realizamos cuando vendemos un contrato que no poseemos, con la intención de comprarlo más barato en un futuro próximo.

2.1.4.2. Salir del Mercado

Una vez hemos obtenido nuestros resultados esperados (beneficios) o bien hemos alcanzado nuestro límite de pérdidas, pasamos a cerrar las posiciones abiertas que tengamos en el mercado, saliendo de este modo del mismo. Al igual que en el apartado anterior, existen dos formas de salir del mercado: cerrar una posición en largo (vender) y cerrar una posición en corto (comprar).

2.2. Competición Robotrader

Para obtener el software con el cual desarrollar nuestro proyecto nos hemos inscrito en el programa RobotraderWorld 2015, evento impartido de forma conjunta por la Universidad Autónoma de Madrid (UAM) y la Universidad Politécnica de Madrid (UPM). En dicho programa se nos concede una licencia de estudiante para poder utilizar los principales softwares de trading algorítmico de la actualidad.

Además, el evento es de ámbito nacional (España) y promueve la colaboración de los estudiantes, por lo que existen una gran variedad de documentación y foros para investigar a fondo desde cualquier parte del país.

Las plataformas de trading disponibles son las siguientes:

- TradeStation (Europed Limited)
- Matlab (MathWorks)
- Darwinex
- Qubitia Solutions
- Trader Workstation (Interative Brokers)
- Ninja Trader

La naturaleza de todas ellas es la misma, gestionar y desarrollar algoritmos de trading que operen de forma automática. Tras estudiar los distintas plataformas, hemos seleccionado el software TradeStation, pues es el que más documentación y foros tiene disponibles, lo que resulta esencial para aprender toda la dinámica de programación necesaria.

3. DISEÑO Y PROGRAMACIÓN DE SISTEMAS

Una vez definidos los conceptos y habiendo seleccionado la plataforma de trading pasamos a la fase de desarrollo de nuestro sistema.

En nuestro caso particular la elección óptima habría sido seleccionar la plataforma Matlab, pues ofrece la mayores posibilidades de programación, pero estamos trabajando sobre un sistema operativo Macintosh (OS X 10.8.5), y todavía no existe versión de Matlab (módulo de trading) adaptada a este sistema. Además, al tratarse de un sistema nuevo en la competición, no existe información de ediciones anteriores que pueda ser recuperada. Por tanto, nuestra elección final ha sido la plataforma TradeStation, para la cual hemos instalado una máquina virtual[6] en nuestro ordenador con el fin de crear una partición dedicada al sistema operativo Windows 7 y trabajar con la plataforma desde ahí.

3.1. ENTORNO DE TRABAJO

La plataforma TradeStation resulta ser la más intuitiva de todas y dispone de un entorno de trabajo ordenado con una gran cantidad de aplicaciones iniciales. El programa permite analizar la serie temporal de un contrato, así como observar barra a barra los resultados de aplicar una determinada estrategia.

Ilustración 3.1 Entorno de trabajo. Fuente: TradeStation. Elaboración propia.

[6] Maquina virtual: Software que permite instalar un sistema operativo dentro de otro sistema operativo, delegando parte del CPU y de la memoria a dicha partición de disco.

TradeStation dispone de un menú general con distintos botones en el panel superior que permiten realizar búsquedas de determinados contratos en un tiempo muy reducido.

Además, permite gestionar todos los ***indicadores***[7] y ***estrategias***[8] mediante accesos directos y/o menús desplegables. Lo que permite operar en el sistema en una sola ventana y comprobar en el gráfico los resultados de nuestro algoritmo sin abandonar la pantalla general. Esto resulta increíblemente útil, ya que toda la información está siempre a la vista, lo que nos permite ser más rápidos y eficientes a la hora de desarrollar el código de programación.

Del mismo modo, TradeStation posee accesos directos que permiten seleccionar los elementos deseados simplemente haciendo un click, lo que facilita el desarrollo de código, la visualización de charts o el análisis de precios de manera inmediata.

Ilustración 3.2. Accesos directos a funciones principales. Fuente: TradeStation. Elaboración Propia.

Del mismo modo, también disponemos de una pantalla que nos permite observar todos los sistemas que hemos desarrollado y activarlos o desactivarlos para ver sus resultados en el entorno gráfico. Dicha ventana será mostrada en la sección 4 del presente trabajo (caso práctico). En dicho entorno podremos realizar cambios para ver las principales diferencias entre los algoritmos y analizar su implicación en el supuesto de que utilicemos más de un sistema a la vez.

[7] Parámetros utilizados para el análisis gráfico (chartista), como pueden ser las bandas de Bollinger.
[8] Se denominan estrategias a los procesos algorítmicos desarrollados en TradeStation.

3.2. DEFINICIÓN Y ESTRUCTURA DEL ALGORTIMO

Un algoritmo es una lista de instrucciones, donde existen una serie de normas sintácticas, mediante la cuales se genera una acción u otra dependiendo de la situación particular con la que se encuentre el sistema. Como resultado, se obtiene una respuesta particular, un output, basado en la información disponible y en cómo dicha información es procesada por los filtros del algoritmo (Fitshchen, 2013).

Los algoritmos en economía y gestión empresarial se utilizan básicamente para: Predecir, Clasificar o Agrupar (Montalvo, 2014). En nuestro caso particular vamos a desarrollar un logaritmo que clasifique la realidad en dos situaciones: **situación A** (entrar al mercado) y **situación B** (salir del mercado). Todo nuestro sistema focalizará en seleccionar los mejores instrumentos o estadísticos (Cáceres, 2009) que nos permitan diferenciar cuál es el mejor momento para entrar o salir del mercado, dando como resultado un algoritmo que ayude a diferenciar entre la situación A y la situación B, para así sacar el máximo beneficio posible.

Resaltar que el objetivo de estas herramientas financieras es nada más y nada menos que facilitar la inversión a través de la aplicación de autómatas que ejecuten operaciones bursátiles, al igual que un operador humano experimentado, pero de forma más veloz.

En nuestro proyecto los algoritmos se realizan bajo el lenguaje de programación EasyLanguague que viene integrado en la plataforma TradeStation. Dicho lenguaje de programación, en adelante código, se estructura según una serie de pautas que el software es capaz de leer. Los elementos más importantes son: los inputs, las variables y declaraciones.

3.2.1. EasyLanguage

Todo software tiene su lenguaje de programación. En el caso de TradeStation, el EasyLanguage es el instrumento mediante el cual podemos comunicarnos con nuestro sistema.

Se trata de un código que el software es capaz de interpretar, y puede escribirse mediante un procesador de texto plano. Puede considerarse como el idioma del programa, ya que mediante este lenguaje explicamos a nuestro sistema qué debe hacer, cómo debe hacerlo y bajo qué circunstancias.

De este modo, EasyLanguage ha sido diseñado específicamente para traders, y muchas de sus palabras reservadas y funciones básicas están denotadas en un lenguaje financiero en inglés muy fácil de comprender. Ésta es una de las principales razones por las cuales seleccionamos esta plataforma de trading. Otro pilar en el que se fundamenta el lenguaje EasyLanguage es su similitud al lenguaje C de programación[9], pues tanto los operadores como las declaraciones condicionales (If, else) son muy familiares.

Asimismo, este lenguaje permite personalizar toda la formulación, a excepción de sus palabras reservadas, lo que abre un mundo de posibilidades para explotar al

[9] Lenguaje informático de programación más utilizado en el mundo.

máximo el software. Esto es exactamente lo que se vamos a realizar en el presente trabajo, pues pasaremos a desarrollar distintas estrategias (algoritmos) para luego poder compararlos entre sí, delimitando de este modo, qué estrategias son más acertadas y por qué.

En definitiva, se trata de un lenguaje sencillo y flexible mediante el cual, no solo podemos personalizar las estrategias de inversión, sino también los indicadores que analicen las señales del mercado. Esta variedad de opciones enriquece la programación, ya que permite utilizar una gran variedad de indicadores, como las bandas de Bollinger[10], y modificarlos según nuestros propios intereses. Por ejemplo, ampliando el número el número de observaciones que son necesarias para el cálculo de las bandas, o estableciendo el número diferente de desviaciones típicas que definan al estadístico.

3.2.2. Inputs y variables

Los **inputs** son las variables de inicialización del sistema, pueden ser denotadas mediante cualquier nombre, a excepción de las **palabras reservadas**[11], y su utilidad radica en que delimitan los valores iniciales de una determinada estrategia haciéndolas factibles y manipulables. En definitiva, su función es la de representar el estado inicial del modelo.

Por tanto, los inputs nos permiten modificar las condiciones del modelo sin necesidad de modificar el resto de la estructura, sino que simplemente basta con sustituir el valor que representa cada entrada (input), de forma que disponemos de una gran flexibilidad a la hora de plantearnos situaciones (TradeStation, 2015). Cada input tiene un nombre único.

A modo de ejemplo:

Para definir nuestro *objetivo de beneficio*[12], podemos declarar un input llamado "objetivo", "beneficio" o cualquier otro nombre que se nos ocurra, siempre que no sea una **palabra reservada**, y asignarle un valor.
Dicho valor puede ser un valor numérico o un parámetro. El EasyLanguague permite diferenciar dos tipos de parámetros: un parámetro boolean[13], o un string[14].
De modo que:

$$Input\colon\ nombreInput\ (valor\ del\ Input)$$

$$Ejemplo \rightarrow\ \ Input\colon objetivo\ (600)$$

En nuestro ejemplo, el valor del input es numérico y representa un objetivo de beneficio de 600 unidades. Es decir, una vez alcanzadas 600 unidades de beneficio el sistema finalizará el contrato.

[10] Indicador de análisis técnico introducido por John Bollinger en 1980. Son dos curvas que envuelven el gráfico y determinan cuando un activo esta sobrecomprado o sobrevendido.
[11] Palabras que se reserva el lenguaje EasyLanguage por ser utilizadas en alguna de sus funciones básicas (if, else, <>…).
[12] Lo que esperamos ganar. Una vez llegamos a dicho punto, finalizamos la operación pues ya hemos obtenido el beneficio deseado. También es denominado profit traget, en inglés.
[13] Un parámetro Boolean representa un sistema de verdadero o falso (true/false) respecto a un determinado input.
[14] Un string representa un texto, el cual puede representar una función matemática.

Por otro lado, las **variables** son una herramienta de programación que permite almacenar valores y hacer referencia a ellas cuando sea necesario. De este modo, las variables permiten organizar el código mediante nombres a nuestra elección, siempre que no sea una palabra reservada, que describan la naturaleza del cálculo o la finalidad de los datos (TradeStation, 2015).

Al igual que los inputs, cada variable posee un nombre único que lo identifica. Además, cada variable declarada debe poseer un valor de partida, el cual generalmente parte de un valor numérico igual a cero (0), pero se puede utilizar cualquier valor, tanto numérico como paramétrico (boolean, strings).

Uno de las principales ventajas de utilizar variables es que podemos sintetizar en una sola palabra una ecuación o conjunto de sucesos declarados, lo que facilita la escritura de código. Además, esto mejora el procesamiento de la información y la velocidad de nuestro autómata, ya que tarda menos tiempo en leer y ejecutar el código del programa (TradeStation, 2015).

Ejemplo:

$$Variable\colon\ nombrevariable\ (valor\ de\ partida)$$

$$Ejemplo \rightarrow\ \ Variable\colon varianza\ (0)$$

En este caso, nuestra variable hace referencia a la varianza, cuya ecuación estará desarrollada en alguna parte de nuestro código. Cada vez que se haga referencia a esta variable el sistema entenderá que hace referencia a dicho elemento y que su valor inicial o de partida es igual a cero.

3.2.3. Declaraciones

Las declaraciones representan la estructura del algoritmo en sí. Son las instrucciones, normas y funciones que configuran todo el conjunto. Nos permiten crear estructuras y conjuntos personalizados a nuestra elección en el sistema.

Para nuestros algoritmos hemos utilizado dos tipos de elementos principales:

- *Conditional Branching*
- *Arrays*

Los ***Conditional Branching*** son estructuras condicionales que realizan una acción u otra dependiendo del resultado.

Son **las estructuras más importantes del sistema**, ya que son las que utilizamos para que nuestro algoritmo tome decisiones de forma autónoma. Hacen referencia a los conocidos arboles de decisión en economía, donde para tomar una decisión debemos tener en cuenta una serie de requisitos.

El más conocido es el comando ***If…Then***, el cual representa, en el lenguaje coloquial, la siguiente estructura: *"Si pasa esto… entonces ejecuta esto"*.

Según TradeStation (2015) *"If statements, are the engines that drive all computer programming languages and applications, they are what computer programming is all about. By performing certain calculations or actions at different times, you are able to build logic that can accomplish almost anything"*.

Conditional Branching	
If… Then	Basada en True/false. *"Si pasa esto… entonces ejecuta esto".*
If… Else…Then	Basada en True/false, pero con alternativa. *"Si pasa esto… entonces ejecuta esto; si **no** pasa esto, ejecuta esto otro".*
If…Then Begin…End	Block Statement: Comienza a hacer algo, pero solo hasta un determinado momento, luego se detiene. *"Si pasa esto… entonces ejecuta esto…hasta que llegue a este límite.*
If…Then Begin…Else Begin…End	*"Si pasa esto… entonces ejecuta esto…hasta que ocurra a esto otro; si **no** pasa esto, ejecuta esto otro… hasta que llegue a este límite".*
Once… Begin… End	*"Una vez que ocurra esto…empieza esto…hasta que llegue a este límite".*

Tabla 3.1. Conditional branching. Fuente: TradeStation (2015). Elaboración Propia.

Por tanto, este tipo de declaraciones son la que estructuran el orden lógico de decisión de todo el sistema. Es decir, son los encargados de reglamentar las normas de actuación del autómata.

Mientras tanto, los ***Arrays*** representan estructuras nuevas que creamos nosotros mismos y pueden ser de cualquier tipo. Suelen utilizarse para crear ecuaciones y definir relaciones entre los distintos inputs.

3.2.4. Operadores

Una vez introducidos los principales elementos de un sistema de trading mediante EasyLanguage debemos comenzar a definir cómo se relacionan dichos elementos. Los operadores son las figuras que definen estas relaciones, y pueden ser dos tipos: lógicos y relacionales.

La importancia de conocer y delimitar unos correctos operadores resulta esencial para un algoritmo eficiente. Ya que representan las relaciones entre los distintos componentes del modelo y cómo reaccionan unos frente a otros.

A continuación mostramos una tabla con los operadores más relevantes ordenados según sus características:

Tipos de operadores		
Operadores	***Función***	***Símbolo***
Lógicos	Y	AND
	O	OR
Relacionales	Igual	=
	Diferente	<>
	Mayor/ menor	>/ <
	Mayor igual / menor o igual	>= / <=
	Cruzar por encima	CROSSES ABOVE
	Cruzar por debajo	CROSSES BELOW

Tabla 3.2. Tipos de operadores. Fuente: TradeStation (2015). Elaboración Propia.

Algunos ejemplos:

$$If\ Close > Close[1]\ then\ Alert;$$

$$If\ Close <>\ Close[1] then\ buy\ this\ bar;$$

En el primer ejemplo estamos programando el algoritmo para que compruebe si el precio de cierre actual es superior (>) al precio de la barra anterior, de ser así se ejecutará una alerta en nuestro sistema informándonos de ello.

Del mismo modo, nuestro segundo ejemplo hace referencia a si los valores son diferentes (<>), en el caso de que sea así, se ejecutará una orden de compra sobre la barra actual.

En definitiva, el uso de los operadores y declaraciones dibujan las líneas de actuación del sistema. De modo que dependiendo de la información, el algoritmo tomará una decisión u otra, entre las cuales puede presentarse la opción de no hacer nada. En concreto, si en alguno de estos dos ejemplos el precio de la acción resulta ser idéntico para ambas barras, el algoritmo no realizará ninguna operación.

4. CASO PRÁCTICO: Desarrollo de Sistemas de Trading

El trading automático es muchas veces similar al arte. Hay veces en las que estarás diseñando una estrategia que nadie podrá entender hasta reflexionarla varias veces, otras son muy detalladas y otras simplemente no tienen sentido. El hecho fundamental, y a la vez denominador común entre arte y trading es que comienzas desde la nada, en blanco.

En definitiva, como toda obra de arte, todo comienzo requiere una inspiración. Nuestra inspiración llegó a raíz de un pequeño libro de la biblioteca de económicas llamado "*Análisis técnico bursátil*" de *Oriol Amat* y *Xavier Puig*, editado en Barcelona en el año 2000. Dicho libro muestra en su apartado 2.7.2 una idea muy simple y a la vez interesante, los *gaps bursátiles*.

A raíz de este esquema, comenzó la fase de programación de sistemas de trading para localizar, evaluar y operar en gaps bursátiles.

4.1. SISTEMAS DE FRECUENCIA DIARIA

4.1.1. Sistema A: gap founder

El presente sistema A constituye el primer algoritmo de nuestro estudio, su estructura es simple, su frecuencia diaria y su funcionamiento sencillo. Para este estudio hemos utilizado la acción Netflix (NFLX) perteneciente al índice NASDAQ (EE.UU.).

```
{Sistema Inicial: algoritmo gap founder}

Inputs:
suelo (500), {stoploss}
cerdito (200); {objetivo de beneficio}

{caso 1, operar en corto}
If Date<>Date [1] and Open>Close [1]
   Then sell short ("Submarino") This bar on close;

{caso 2, futura subida}
If date<>date[1] and open<close[1]
   Then buy ("Cohete") This bar on close;

Setstoploss (suelo);
Setprofittarget (cerdito);
Setexitonclose;
```

Ilustración 4.1. Sistema A: gap founder. Fuente: TradeStation. Elaboración propia.

Siguiendo a Little (2013), se trata de un algoritmo con dos inputs numéricos, denominados *suelo* y *cerdito*. El primer input hace referencia al *Stoploss* del sistema, que hace referencia a una pérdida de 500 puntos. Por otro lado, el segundo input hace referencia al *objetivo de beneficio*, establecido en 200 puntos.

Como podemos observar no existen variables en el modelo. Solo existen dos declaraciones condicionales (caso1 y caso2), las cuales establecen las reglas de entrada del sistema en el mercado.

El procedimiento es sencillo, si se dan las condiciones del sistema (caso1, caso2), el autómata ejecutará la operación correspondiente y entrará en el mercado (situación A[15]) adquiriendo o vendiendo en corto un contrato. Y no saldrá del mismo hasta que se alcance una de las tres posibles situaciones:

1) Que se alcance el objetivo de beneficio de 200 puntos.
   (función: ***setprofittarget***)
2) Que se sufra una pérdida de 500 puntos.
   (función: ***setstoploss***)
3) Que no se cumplan 1) y 2), entonces cerrará el contrato al final de la sesión.
   (función: ***setexitonclose***)

Caso1 (submarino):

Si el valor del parámetro día de la barra anterior (date[1]) es diferente (<>) del parámetro día de la barra actual (date), ***y*** *si el valor de apertura de hoy (open)* ***es mayor*** *que el valor de cierre de ayer(close[1]), entonces* ***vender en corto (sell short).***

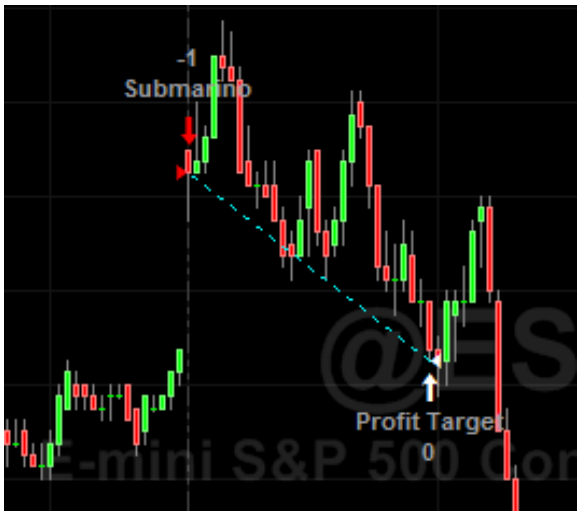

Ilustración 4.1.1. Submarino con ganancias.

Ilustración 4.1.2. Submarino con pérdidas.

Fuente: TradeStation. Elaboración Propia.

La condición del caso1 permite entrar en el mercado cuando se detecta un gap bajista. Es decir, cuando suponemos que el precio de apertura del día de hoy está sobrevalorado y va a bajar a lo largo de la sesión.

En la ilustración 4.1.1. podemos observar este escenario, pues se puede apreciar un gap entre el valor de cierre del día anterior y el valor de apertura del día siguiente (momento donde se ejecuta la orden "submarino"). Como se observa en dicho gráfico, al cabo de un tiempo la cotización vuelve a su valor tendencial lo que permite que el gap se cierre y la estrategia obtenga beneficios. En la ilustración 4.1.2. el gap no llega a estabilizarse y la estrategia cierra con pérdidas.

[15] Referencia al apartado *3.2 Definición y Estructura del algoritmo,* de este proyecto.

Caso2 (cohete):

Si el valor del parámetro día de la barra anterior (date[1]) es diferente (<>) del parámetro día de la barra actual (date), ***y*** *si el valor de apertura de hoy (open)* ***es menor*** *que el valor de cierre de ayer(close[1]), entonces* ***comprar (buy).***

Ilustración 4.1.3. Cohete con beneficios.

Ilustración 4.1.4. Cohete con perdidas.

Fuente: TradeStation. Elaboración propia.

La condición del caso2 permite entrar en el mercado cuando se detecta un gap alcista. Es decir, cuando suponemos que el precio de apertura del día de hoy está por debajo de su valor tendencial y que por lo tanto va a subir a lo largo de la sesión.

Resultados del Sistema A: gap founder (a 2 años[16])			
	ALL trades	LONG trades[17]	SHORT trades[18]
Beneficio bruto	$73.810,00	$32.467,00	$41.343,00
Pérdidas brutas	($61.047,00)	($28.947,00)	($32.100,00)
Beneficio neto[19]	**$12.763,00**	**$3.520,00**	**$9.243,00**

Tabla 4.1. Resultados del sistema A: gap founder. Fuente: TradeStation. Elaboración propia.

Como podemos observar en la tabla superior, mediante este sistema hemos obtenido unos resultados positivos.

Según Ajram (2013), la justificación de este escenario cíclico radica en el aumento del volumen de negociación durante los primeros minutos de cotización bursátil. Dicho incremento, puede generar tensiones alcistas o bajistas que dan lugar a la aparición de estos gaps, y que pasados unos minutos van desapareciendo, recuperando de nuevo la cotización el precio de cierre del día anterior.

[16] Para calcular los resultados se ha realizado un *Backtesting* a dos años atrás. (03 Junio 2013 - 01 Junio 2015)

[17] Operaciones realizadas en largo, caso 2.

[18] Operaciones realizadas en corto, caso 1.

[19] Beneficio neto = Beneficio bruto – Pérdidas brutas.

Por tanto, nuestro algoritmo aprovecha estos cambios tendenciales para *vender en corto/comprar* cuando el precio esta *sobrecomprado/sobrevendido*, y luego *comprar para cubrir/vender* cuando el precio *baja/sube*. Obteniendo de este modo un beneficio.

Recordar que según indicamos en la introducción de este proyecto, trabajamos bajo condiciones ideales de mercado [20] sin tener en cuenta los efectos de deslizamiento[21]. Lo que se traduce como una bondad de ajuste[22] perfecta entre los valores esperados y los observados (Cáceres, 2007).

4.1.2. Sistema B: gap founder plus

En este caso, el sistema B resulta ser una versión ampliada del sistema A, la modificación realizada en este sistema persigue mantener un mayor control sobre la variabilidad. Para ello, se añaden **dos condiciones auxiliares** a las declaraciones caso1 y caso 2.

```
{Sitema inicial con HighD y LowD: algoritmo gap founder plus}

Inputs:
suelo (500), {stoploss}
cerdito (200); {objetivo de beneficio}

{caso 1, operar en corto}                          Condición Aux. 1
If Date<>Date [1] and Open>Close [1] and Open<HighD (1)
   Then sell short ("Submarino") This bar on close;

{Caso 2, futura subida}                            Condición Aux. 2
If date<>date[1] and open<close[1] and open>lowd (1)
   Then buy ("Cohete") This bar on close;

Setstoploss (suelo);
Setprofittarget (cerdito);
Setexitonclose;
```

Ilustración 4.2. Sistema B: gap founder plus. Fuente: TradeStation. Elaboración propia.

El procedimiento es idéntico al sistema A, pero en este caso la volatilidad disminuye a causa de esta nueva formulación.

La función *HighD (1)* representa el valor más alto en la sesión de ayer, mientras que el *LowD (1)* representa justo lo contrario, el valor más bajo (TradeStation, 2015). Al preguntarnos si el precio de apertura actual ha sobrepasado los máximos/mínimos del día anterior nos proporciona información sobre su volatilidad. Lo que nos es verdaderamente útil para representar la variabilidad en nuestro algoritmo.

---

[20] Situación hipotética de que no existan intermediarios entre el sistema y el mercado real, es decir, sin comisiones añadidas al coste de operación y con un tipo de interés del 2%.

[21] Efecto deslizamiento (*Slippage*): Es el efecto producido por cambios bruscos en el volumen de negociación, lo que provoca que el precio al que se decide vender un contrato no coincida con el precio de venta.

[22] Bondad de ajuste: Estadístico que representa la calidad predictiva de un modelo, en nuestro caso, que la orden de operación sea igual al resultado de operación.

El concepto en cuestión puede parecer engorroso. La idea básica es que si nuestro contrato, en este caso la acción de Netflix (NFLX), aparece con un precio de apertura superior o inferior a los máximos registrados ayer, significa que el valor del contrato no está en sus niveles normales, probablemente ha llegado información nueva al mercado, y la cotización está lejos de su valor tendencial.

Resultados del Sistema B: gap founder plus (a 2 años)			
	ALL trades	LONG trades	SHORT trades
Beneficio bruto	$57.402,00	$26.259,00	$31.143,00
Pérdidas brutas	($41.373,00)	($21.777,00)	($19.596,00)
Beneficio neto	**$16.029,00**	**$4.482,00**	**$11.547,00**

Tabla 4.2. Resultados sistema B: gap founder plus. Fuente: TradeStation. Elaboración propia.

Los resultados netos de este sistema son mejores que el anterior. En concreto 3.266 dólares más de beneficio neto que su sistema homólogo. Pero aún así, no solo nos interesa el resultado contable. Ya que aún teniendo mayores beneficios netos, el sistema B obtuvo menos beneficios brutos y menos pérdidas brutas que el sistema simplificado (sistema A), debido a las exigencias adicionales que hacen que el sistema tome menos riesgos y seleccione mejor las operaciones en las que participa. Esta revisión, junto al resto de parámetros se detallará en la sección 4.1.4 de este proyecto, donde compararemos los sistemas.

4.1.3. Sistema C: gap Bollinger

Una vez hemos explorado cómo evoluciona un sistema con parámetros numéricos, pasamos a desarrollar un sistema en el cual las variables del modelo vayan cambiando según cambian las circunstancias. Para ello, hemos decidido utilizar el indicador de las ***bandas de Bollinger***. Según Chica (2015), es uno de los estimadores básicos de un sistema de trading, pues evalúa la variabilidad de la serie a raíz de un número concreto de observaciones pasadas. Para ello, utiliza dos bandas que calcula mediante desviaciones típicas a ambos lados de los datos, por debajo y por encima de la serie temporal.

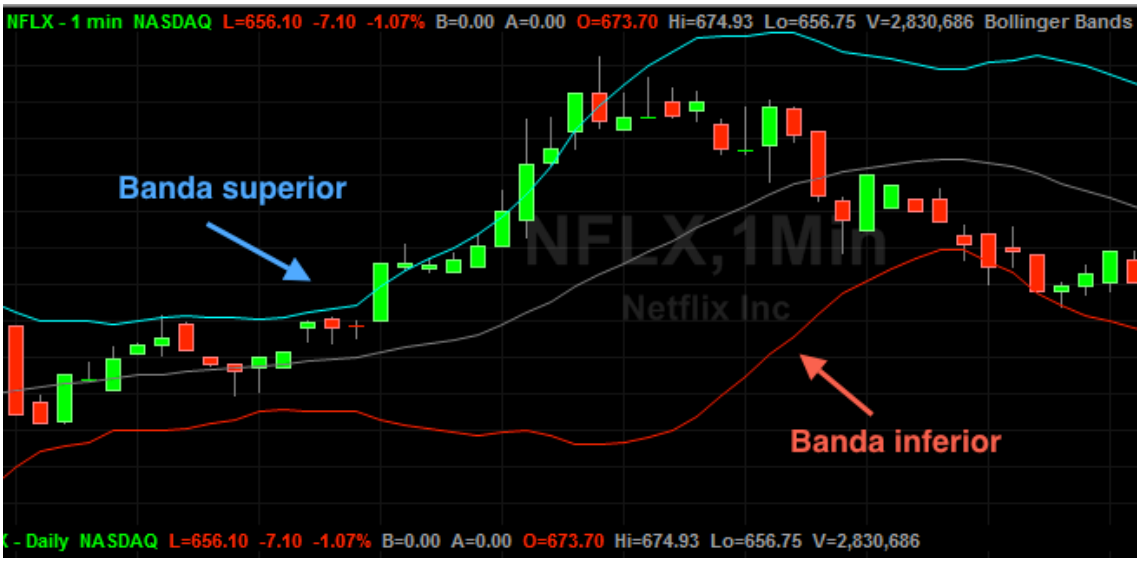

Ilustración 4.3. Bandas de Bollinger. Fuente: TradeStation. Elaboración Propia.

El efecto es similar a los indicadores HighD y LowD, pero esta vez se puede observar directamente en el gráfico. Por tanto, cuando un punto de la serie se encuentre por encima de la barra superior de Bollinger, el contrato estará sobrecomprado, por lo que deberíamos iniciar una venta en corto, pues es muy probable que el valor no siga subiendo y vuelva a estabilizarse. Lo mismo ocurre cuando el valor de cotización supera la banda inferior, solo que su efecto es el contrario y nos interesa comprar, ya que es muy probable que su valor suba de nuevo.

Las bandas de Bollinger son como unas resistencias que modulan la serie, cuando la cotización sobrepasa dichos márgenes el algoritmo determina que el precio del activo está por encima o por debajo de su valor tendencial.

Para desarrollar esta estrategia de forma correcta hemos tenido que dividir el algoritmo en dos códigos diferentes. Uno para operar en corto (gaps bajistas) y otro para operar en largo (gaps alcistas).

```
{Buscador de gaps bajistas mediante bollinger}

Inputs:
    double BollingerPrice( Close ),
    int Length( 20 ),{Número de barras a tomar cuenta para calcular
                      las bandas de bollinger}
    double NumDesvUp( 2 ),{número de desviciones típicas hacia arriba}
    double NumDesvDwn( -2 );{número de desviciones típicas hacia arriba}

variables:
    double Avg( 0 ),
    double Desv( 0 ),
    double Bandainferior( 0 ),
    double Bandasuperior( 0 );

Avg = AverageFC( BollingerPrice, Length ) ;
Desv = StandardDev( BollingerPrice, Length, 1 ) ;
Bandasuperior = Avg + NumDesvUp * Desv ;
Bandainferior = Avg + NumDesvDwn * Desv ;

{vende caro para comprar barato}
If Date<>Date [1] and Open>Close [1]
   Then sell short ("Submarino")This bar on close;


Setstoploss (Bandasuperior);
Setprofittarget (Bandainferior);
```

Ilustración 4.3.1. Sistema C: gap Bollinger submarino. Fuente: TradeStation. Elaboración Propia.

En este sistema podemos observar como la estructura es más compleja. Utilizamos dos tipos de inputs y variables, ***double*** permite hacer referencia a valores reales positivos o negativos, lo que permite delimitar ambas bandas, e ***int*** que hace referencia a números enteros.

Seguidamente, podemos observar cuatro inputs: ***BollingerPrice*** hace referencia al precio de estudio, es decir los valores de cotización de la serie; ***Length*** representa el

número de barras (n) a tomar en cuenta; ***NumDesvUp*** y ***NumDesvDwn*** son el número de desviaciones típicas, hacia arriba y hacia abajo respectivamente, que utilizará el algoritmo para dibujar las bandas.

Respecto a la variables, tenemos definida la desviación típica (*Desv*), la media (*Avg*) y las dos bandas. Todos con un valor inicial de cero.

Para calcular el valor de la variables, el algoritmo recurre a las declaraciones expuestas en el código. Por ejemplo, para hallar la media (*Avg*) disponemos de la función ***AverageFC***, la cual toma en consideración los valores de cotización (*BollingerPrice*) y las n observaciones a tomar en cuenta (*Length*).

Lo mismo ocurre con la desviación típica ***StandardDev*** y las otras dos declaraciones, ***Bandasuperior*** y ***Bandainferior,*** son las fórmulas que determinan el valor de los parámetros del sistema.

Por último, tenemos la misma estructura (caso1, caso2) que rige el orden de entrada en el mercado. Pero esta vez, al disponer de un indicador más complejo como son las bandas de Bollinger se utilizan dichos parámetros para salir del mercado. Además, al controlar la variabilidad mediante este sistema de bandas, no es necesario integrar las condiciones auxiliares del sistema B en este código.

```
{Buscador de gaps alcistas mediante bollinger}

Inputs:
     double BollingerPrice( Close ),
     int Length( 20 ),{Número de barras a tomar cuenta para calcular
                       las bandas de bollinger}
     double NumDesvUp( 2 ),{número de desviciones típicas hacia arriba}
     double NumDesvDwn( -2 );{número de desviciones típicas hacia arriba}

variables:
     double Avg( 0 ),
     double Desv( 0 ),
     double Bandainferior( 0 ),
     double Bandasuperior( 0 );

Avg = AverageFC( BollingerPrice, Length ) ;
Desv = StandardDev( BollingerPrice, Length, 1 ) ;
Bandasuperior = Avg + NumDesvUp * Desv ;
Bandainferior = Avg + NumDesvDwn * Desv ;

{Compra barato para vender caro}
If date<>date[1] and open<close[1]
   Then buy ("Cohete")This bar on close;

Setstoploss (Bandainferior);
Setprofittarget (Bandasuperior);
```

Ilustración 4.3.2. Sistema C: gap Bollinger cohete. Fuente: TradeStation. Elaboración propia.

Resultados del Sistema C: gap Bollinger (a 2 años)			
	ALL trades	LONG trades	SHORT trades
Beneficio bruto	$96.075,00	$44.105,00	$51.970,00
Pérdidas brutas	($85.102,00)	($36.863,00)	($48.239,00)
Beneficio neto	**$10.973,00**	**$7.242,00**	**$3.731,00**

Tabla 4.3. Resultados sistema C : gap Bollinger. Fuente: TradeStation. Elaboración propia.

4.1.4. Comparaciones entre modelos y No Free Lunch Theorem

Una vez definidos los tres sistemas vamos a compararlos entre sí para ver cuáles son sus principales diferencias. Según Wolpert (1997), el ***"No Free Lunch Theorem"*** defiende que no existe un sistema perfecto, sino que cada sistema debe construirse alrededor de una situación concreta, para así ser de utilidad y llegar a ser eficiente.

Por tanto, la búsqueda del *Santo Grial* de los sistemas de trading automáticos queda descartada, pues cada sistema está adecuado a su entorno, no existe un sistema que sea polivalente. Un sistema con muy buenos resultados en un determinado contexto puede pasar a ser el peor si las condiciones de mercado cambian.

Resultados sistemas diarios (a 2 años)			
	Sistema A: *gap founder*	***Sistema B:*** *gap founder plus*	***Sistema C:*** *gap Bollinger*
Beneficio bruto	$73.810,00	$57.402,00	$96.075,00
Pérdidas brutas	($61.047,00)	($41.373,00)	($85.102,00)
Beneficio neto	**$12.763,00**	**$16.029,00**	**$10.973,00**
Número de contratos ejecutados	378	502	483
Porcentaje de acierto (Win rate)	73,90%	76,46%	54,66%
Beneficio medio por contrato	$25,42	$42,40	$22,72
Mayor ganancia	$209,00	$209,00	$1.255,00
Mayor pérdida	($512,00)	($512,00)	($1.337,00)
Tiempo en el mercado	1,95%	1,50%	14,29%
Tiempo medio de operaciones abiertas[23]	40 Minutos	41 Minutos	5 horas, 10 minutos
Max. Drawdown	($4.018,00)	($3.000,00)	($6.433,00)

Tabla 4.4. Resultados sistemas diarios. Fuente: TradeStation. Elaboración propia.

Por debajo y por encima de la serie temporal.
un contrato y su venta.

4.1.4.1. Preferibilidad entre modelos

Siguiendo a Wolpert (1997) y Wolpert (2005), podemos basar el orden de preferibilidad en la eficiencia, siempre y cuando se evalúen en un mismo entorno.

Debemos tener presente que el orden de preferencia puede modificarse a raíz de cambios en el entorno. Aun así, pueden existir modelos que son siempre preferibles a otros, idea contraria a la doctrina *No Free Lunch*. Este hecho se da cuando comparamos sistemas similares con estimadores distintos.

Dicha apreciación se ve reflejada en los sistema A y B, donde el sistema *gap founder plus (B)* es siempre preferible al *sistema simplificado (A)*. La causa es que ambos representan un mismo proceso (misma estructura logarítmica), pero no poseen los mismos estadísticos, de modo que unos son más eficientes que los otros (Cáceres, 2009). Esto permite que uno de los modelos obtenga siempre unos resultados más favorables.

En nuestro estudio, dichos estadísticos corresponden a las funciones auxiliares del *sistema B*, definidas en el apartado 4.1.2 de este proyecto. Ya que permiten errar menos y obtener unos beneficios netos superiores.

Por otro lado, podemos observar el caso contrario mediante los modelos B y C, que utilizan estructuras algorítmicas distintas. Es decir, diferentes procesos para analizar y operar en la serie, lo que provoca que obtengan unos resultados dispares en casi todos sus parámetros. A modo de ejemplo, el *win rate*[24] del *sistema B* es superior a *C*, pero al mismo tiempo sus beneficios brutos son inferiores. Ante estas situaciones nos planteamos preguntas del estilo: *¿Qué vale más? ¿Seguridad o beneficio?*

Responder a este tipo de preguntas dista mucho del mero análisis cuantitativo, pues intervienen más las razones estratégicas en base a los distintos perfiles de riesgo del inversor.

Finalmente, ésta es la razón por la cual nuestro énfasis en el concepto de *No Free Lunch* sea elevado. Ya que un sistema cuantitativo no sirve de mucho si no persigue unos objetivos discrecionales concretos. Lo cual es la causa de que no pueda existir un sistema perfecto, pues los sistemas se hacen a medida.

4.1.4.2. La cuestión temporal

A continuación, centramos nuestra atención en las diferencias entre el sistema B y el sistema C. La principal diferencia entre *founder plus* y *Bollinger* es que los parámetros de salida del primero son fijos (numéricos), mientras que en el segundo dichos valores son determinados por una función y pueden variar. Bajo esta diferencia radican los resultados tan dispares entre los dos sistemas, pues mientras uno resulta ser más estricto, el otro permite una mayor flexibilidad.

Como podemos observar, el tiempo que pertenece el sistema *founder plus* dentro del mercado[25] ronda el 1,5% del total de dos años. En cambio, el sistema *Bollinger* opera el 14,29% del mismo. Lo cual refleja una mayor exposición del segundo respecto

[24] $win\ rate = \left(\frac{Ganadoras}{Total}\right) * 100$

[25] Con posiciones abiertas. Es decir, con contratos en su poder que desea cerrar.

al primero. La causa es que los parámetros del algoritmo *Bollinger* referentes a la salida del mercado están denotados por funciones variables, cambiantes con el mercado, y no por valores prefijados.

Del mismo modo, las operaciones tienen una duración media de cuarenta minutos en el sistema *founder plus* frente a unas cinco horas y diez minutos del sistema *Bollinger*. Este dato nos sirve para confirmar la mayor exposición del sistema de bandas, pues permanece mayor tiempo en el mercado y soporta en mayor medida las variaciones en la serie.

A modo de ejemplo, si en un determinado momento la volatilidad del mercado aumenta , el sistema *Bollinger* realizará sus cálculos y detectará que la variabilidad es más elevada, pero permanecerá en el mercado si no se llega a los límites de las bandas. Sin embargo, el algoritmo *founder plus* no detectará este cambio, ya que tiene unos valores de referencia fijos. En definitiva, *founder plus* es menos flexible, lo que provoca que cierre las operaciones y salga del mercado con más facilidad ante cualquier cambio en la serie.

En conclusión, la cuestión temporal en estos dos modelos determina su comportamiento, por lo que dependiendo de nuestros objetivos seleccionaremos una característica (flexibilidad) u otra (rigidez).

4.1.4.3. Estabilidad y eficiencia

Según la tabla 4.4. los resultados del sistema *founder plus* son preferibles a los del sistema *Bollinger*. Sin embargo, este proyecto no busca analizar cuál es el mejor modelo para un periodo concreto, sino estudiar cómo estructurar un buen sistema acorde a cualquier situación que se nos presente.

En este caso particular, para este valor concreto (NFLX) y periodo de tiempo específico y frecuencia, el ganador a efectos contables es el *gap founder plus*. Pero a lo que respecta a este estudio, el sistema preferido es *el gap Bollinger.*

Atendiendo a los datos, *founder plus* aglutina 5.056 dólares más que su homólogo *Bollinger*. Sin embargo, el sistema *Bollinger* acumula 38.673 dólares más de beneficio bruto. Es decir, no existe una continuidad entre quien consigue más y quien gana más, para ello hemos desarrollado el siguiente indicador (*WouldWin*) que mide la ganancia potencial del sistema según lo que podría haber ganado y lo que realmente gana.

De modo que:

$$WouldWin = \frac{Beneficios\ brutos}{Beneficios\ netos} = Participación\ sobre\ beneficios\ netos$$

Resultados del indicador *WouldWin*			
	gap founder	gap founder plus	gap Bollinger
WouldWin	5,78	3,58	8,75

Tabla 4.5 Resultados indicador WouldWin. Fuente: TradeStation. Elaboración propia.

Este parámetro representa la ganancia potencial del sistema, mide la divergencia entre beneficios brutos y beneficios netos. Como podemos apreciar, el sistema *Bollinger* podría haber ganado 8,75 veces más de lo que se realmente ha ganado (beneficio neto). Esto es debido a que ha sufrido más pérdidas brutas que el resto de sistemas, pero aún así, es el que mayor beneficios brutos ha obtenido. Por lo que si estas pérdidas disminuyeran el beneficio neto podría aumentar hasta 8,75 veces más, de ahí el nombre del indicador.

A efectos contables, el sistema *founder plus* es el que mejores resultados ha obtenido debido a su configuración de restricciones auxiliares, que dan más estabilidad al sistema. Sin embargo, estas restricciones limitan su efectividad, representada como beneficio bruto posible. Esta idea se ve reflejada en Koch (2003), que señala que el beneficio es directamente proporcional al riesgo. El indicador *WouldWin* del sistema *founder plus* representa que su beneficio hubiese sido 3,58 veces superior de no haber sufrido pérdidas, cifra inferior a la que se hubiera obtenido el sistema *Bollinger*.

Del mismo modo, el *gap founder* simplificado ha obtenido peores resultados contables que el *sistema plus*. Pero ha obtenido una mayor capacidad de beneficio (5,78), solo que las pérdidas lo han mermado. La causa es su exposición al riesgo, que es más pronunciada que el sistema *plus.*

A modo de conclusión, no solo debemos dejarnos llevar por los resultados contables, sino también tener consciencia sobre cómo llegamos a dichos resultados. Ya que si la serie analizada fuese más agresiva, con mayor variabilidad o con fuertes cambios de tendencia, el *sistema Bollinger* sería sin lugar a dudas el que mayores beneficios obtendría. Pues sería el que mejor se adaptaría a la inestabilidad.

4.2. SISTEMAS DE FRECUENCIA INTRADIARIA

Las tecnologías computacionales han creado una nueva forma de hacer las cosas debido a su elevada eficiencia, sobre todo si son tareas repetitivas.

Por tanto, no es de extrañar que estas nuevas herramientas generen cambios en los mercados financieros, pues hoy día los ordenadores son capaces de gestionar una ingente cantidad de información y una rápida toma de decisiones. Esto ha generado una nueva forma de operar denominada High-Frecuency-Trading (HFT) lo que constituye una rama propia de las finanzas, donde los ordenadores ocupan el eje central.

Mientras que con los gaps diarios se intentaba buscar un autómata que emulara las acciones de un operador humano de forma automática, en este apartado se desarrolla un algoritmo que opera con mucha mayor frecuencia que los sistemas presentados anteriormente.

4.2.1. Sistema D: *Glory madness*

Para ello, hemos dejado a un lado el marco de los gaps bursátiles y hemos desarrollado un algoritmo que se centre en buscar beneficios ínfimos en cualquier tramo de la serie. Citando a Fraga (2012), pasamos del ***trading discrecional***, donde desarrollamos una estrategia basada en un objetivo concreto y estudiado, al ***trading sistemático***, donde no se persigue un objetivo concreto que hemos analizado, sino que establece un sistema de órdenes que buscan beneficios, por muy pequeños que sean, todo suma. En definitiva, el código que hemos desarrollado es el siguiente:

```
{Buscador de gloria}

Inputs:
    double BollingerPrice( Close ),
    int Length( 20 ),{Número de barras a tomar cuenta para calcular
                      las bandas de bollinger}
    double NumDesvUp( 2 ),{número de desviciones típicas hacia arriba}
    double NumDesvDwn( -2 );{número de desviciones típicas hacia arriba}

variables:
    double Avg( 0 ),
    double Desv( 0 ),
    double Bandainferior( 0 ),
    double Bandasuperior( 0 );

Avg = AverageFC( BollingerPrice, Length ) ;
Desv = StandardDev( BollingerPrice, Length, 1 ) ;
Bandasuperior = Avg + NumDesvUp * Desv ;
Bandainferior = Avg + NumDesvDwn * Desv ;
Value1 = PercentChange(Close,1);
Value2 = Value1-Value1[1];

if Value2 < 0 then Buy ( "HARD" ) this bar on close ;

if MarketPosition > 0 then Sell ( "OUT" ) this bar on close;

Setstoploss (Bandainferior);
Setprofittarget (Bandasuperior);
```

Ilustración 4.4. Sistema D: Glory Madness. Fuente: TradeStation. Elaboración propia.

Este código representa el Sistema D: Glory madness, denominado en la fase de desarrollo como "*Buscador de gloria",* por los altos beneficios que se alcanzaron.

Como se puede observar, hemos exportado la estructura algorítmica de las bandas de Bollinger para este sistema. Como vimos en el apartado anterior, las bandas no son el mejor sistema para obtener beneficios netos, pero sí es el que mayor tiempo mantiene las operaciones en el mercado. Este hecho encaja muy bien con este tipo de algoritmos sistemáticos, ya que ahora nuestro interés no es acertar el tiro, sino errar lo menos posible mientras usamos una ametralladora.

Ilustración 4.5. Diferencia algoritmo discrecional y sistemático.

Como podemos apreciar en las imágenes, un algoritmo sistemático aglomera cada franja temporal mediante operaciones. A ser la frecuencia en minutos, observamos cómo se opera prácticamente en toda la serie buscando beneficios ínfimos. Pero gracias al gran volumen de los mismos, al sumarse generan unos cuantiosos beneficios.

Asimismo, el *No Free Lunch Theorem* queda latente tras esta imagen. Pues se han aplicado dos algoritmos distintos a una misma serie, obteniendo resultados favorables en ambos casos, pero totalmente distintos.

Respecto a las declaraciones del algoritmo, están diseñadas para buscar cualquier tipo de beneficio. Se define una la variable 1 (value1) denotada por la función ***PercentChange***, que representa el cambio porcentual entre el precio de cierre actual (valor) y un precio de n observaciones atrás:

$$PercentChange\ (Valor, n)$$

En este caso, n es igual a uno. Por tanto, la función analiza el cambio porcentual entre el precio de cierre actual respecto al su momento anterior (n=1). Como la frecuencia es en minutos, esto quiere decir que analiza el cambio porcentual del precio minuto a minuto.

Seguidamente, definimos la variable 2 (value2), que analiza la diferencia entre el value1 actual y su respectivo value1 anterior. En este caso representa el minuto anterior. Si la diferencia resulta ser negativa (< 0), significa que el contrato ha bajado de precio, por lo que se ejecuta la orden de compra (HARD). Luego, vendemos el contrato cuando obtengamos un beneficio, el cual queda representado por la declaración con la función ***MarketPosition***.

Para finalizar, hemos mantenido las funciones ***Setstoploss*** y ***Setprofittarget*** mediante las bandas de Bollinger para poseer parámetros de estabilización en el sistema en caso de que se produzcan cambios muy bruscos en la cotización. Este hecho suele ser muy común, debido a que al realizar operaciones minuto a minuto los precios pueden variar muy rápido desde el momento de compra. De este modo, las bandas ayudan a dar estabilidad al sistema.

Resultados del Sistema D: Glory madness (a 2 años)			
	ALL trades	LONG trades	SHORT trades
Beneficio bruto	$1.418.610,00	$1.418.610,00	$0,00
Pérdidas brutas	($1.228.907,00)	($1.228.907,00)	$0,00
Beneficio neto	$189.703,00	$189.703,00	$0,00

Tabla 4.6. Resultados sistema D: Glory madness. Fuente: TradeStation. Elaboración propia.

Los resultados resultan llamativos, pues el procedimiento ha sido relativamente sencillo y los resultados muy boyantes. Sin embargo, no ha sido la complejidad del sistema la que nos ha otorgado dicho beneficios, sino el procedimiento.

El rápido procedimiento de análisis y compra-venta, denominado High-frecuency-trading (HFT), es lo que ha generado dichos beneficios. Pues se aprovecha de su gran velocidad para captar márgenes por operación que le resultan imposibles a un trader convencional.

4.3. PORTAFOLIO DE SISTEMAS

Un portafolio de sistemas de trading es, según Pla (2013), un conjunto de sistemas que funcionan de forma coordinada. Esta aplicación se utiliza cuando queremos alcanzar un objetivo final que abarca muchos componentes, lo que requiere programar más de un sistema para alcanzar los resultados esperados.

Sin embargo, como bien señala Burns (2003), el análisis cuantitativo en series temporales tiene características caprichosas. Pues dos elementos que de forma general dan resultados positivos, pueden pasar a dar pérdidas en el ámbito de las series temporales. Todo gracias al efecto del tiempo.

En nuestro caso práctico hemos desarrollado dos tipos de sistemas, uno diario; que localiza y opera los gaps bursátiles; y uno intradiario, que se encarga de localizar y operar posibles beneficios mediante el cálculo de cambios porcentuales.

Incluimos por tanto en nuestro portafolio dichos algoritmos, para luego proceder al estudio de los mismos. Asimismo, al tener dos variantes claras en nuestro sistema diario (*founder y Bollinger*), desarrollaremos dos portafolios diferentes, uno con el sistema *founder plus* y otro con *las bandas de Bollinger* para analizar si ambos llegan a resultados similares.

4.3.1. Portafolio A: *founder plus + madness*

Este primer portafolio aglutina el algoritmo diario *gap founder plus* y el algoritmo intradiario *glory madness*. La relación de sistemas en TradeStation queda de la siguiente forma:

	Name	Input Values	Status	Buy	Sell	Sell Short	Buy to Cover
	Erredelugo_madness_gap	Close,20,2,-2	On	On	On		On
	Gap_Erredelugo_highD_lowD	500,200	On	On	On	On	On

Ilustración 4.6. Esquema de sistemas portafolio A. Fuente: TradeStation. Elaboración Propia.

La estrategia "*Gap_Erredelugo_HighD_lowD"* hace referencia al sistema *gap founder plus*, se ha denotado de esta forma para hacer mención a sus restricciones auxiliares *(HighD, LowD*), mientras que "Erredelugo_madness_gap" equivale al sistema glory madness. Los nombres son estrictamente organizativos, no representan ningún otro atributo. Además, se ha utilizado la denominación "Erredelugo" en ambos algoritmos debido a que es mi nombre de usuario en el software TradeStation.

Resultados portafolio A: gap founder plus + madness (a 2 años)			
	ALL trades	LONG trades	SHORT trades
Beneficio bruto	$1.372.696,00	$1.363.432,00	$9.264,00
Pérdidas brutas	($1.215.791,00)	($1.206.431,00)	($9.360,00)
Beneficio neto	$156.905,00	$157.001,00	($96,00)

Tabla 4.7. Resultados portafolio A. Fuente: TradeStation. Elaboración propia.

En base a los resultados obtenidos podemos apreciar que aplicar el portafolio es menos rentable que aplicar el Glory madness gap de forma individual.

Esto es debido a la mala gestión de efectivo (money management) que sufre el portafolio. Tanto Fitshchen (2013) como Burns y Wellings (2003) señalan esta idea, la cual quiere decir que si disponemos de una determinada cantidad en efectivo[26], para ejecutar operaciones procedentes de dos sistemas diferentes al mismo tiempo. Se dispondrá, en términos medios, de menos capital para invertir en cada sistema. En otras palabras, los dos sistemas pelearán por el capital.

Por otro lado, esto no ocurriría si tuviésemos dos cuentas independientes. Cada una de ellas asociada a un solo algoritmo, por lo que las operaciones no se entorpecerían unas a otras.

4.3.2. Portafolio B: *Bollinger + madness*

A través de este segundo portafolio unificamos el algoritmo diario *gap Bollinger,* subdividido en dos partes, tal y como se definió en el apartado 4.1.3 de este proyecto, junto al algoritmo intradiario *glory madness*. La relación de sistemas en TradeStation queda de la siguiente forma:

	Name	Input Values	Status	Buy	Sell	Sell Short	Buy to Cover
	Gap_Cohete_Bollin_Erredelugo	Close,20,2,-2	On	On	On		On
	Gap_Submarino_Bollin_Erredelugo	Close,20,2,-2	On		On	On	On
	Erredelugo_madness_gap	Close,20,2,-2	On	On	On		On

Ilustración 4.7. Esquema de sistemas portafolio B. Fuente: TradeStation. Elaboración Propia.

La columna *"Status"* informa si el algoritmo en cuestión se encuentra en funcionamiento. En este primer análisis observamos que los tres elementos se encuentran activos ("*On"*).

Asimismo, la estrategia *"Gap_Cohete_Bollin_Erredelugo"* junto a la estrategia *"Gap_Submarino_Bollin_Erredelugo"* representan el sistema C (*gap Bollinger*). Mientras tanto, al igual que el portafolio A, "Erredelugo_madness_gap" equivale al sistema *glory madness.*

Resultados portafolio B: gap Bollinger + madness (a 2 años)			
	ALL trades	LONG trades	SHORT trades
Beneficio bruto	$1.429.489,00	$1.417.782,00	$11.707,00
Pérdidas brutas	($1.240.606,00)	($1.228.032,00)	($12.574,00)
Beneficio neto	$188.883,00	$189.750,00	($867,00)

Tabla 4.8. Resultados portafolio B. Fuente: TradeStation. Elaboración propia.

[26] En nuestros sistemas contamos con un capital de 1 millón de euros.

Extraordinariamente, en esta ocasión tampoco se consigue superar al sistema glory madness operando de forma individual. En este punto es donde recurrimos de nuevo a Burns y Wellings (2003), pues el efecto del tiempo ha provocado que las operaciones en corto representen pérdidas. Este paradójico efecto ha sido causado por la implementación simultánea de los dos sistemas.

Ante las pérdidas obtenidas en corto, se decide no aplicar el algoritmo *"Submarino"* en este portafolio. Para ello, aprovechando que el código está dividido en dos algoritmos, únicamente debemos cambiar la casilla de "Status" a modo *"Off"*.

De modo que:

	Name	Input Values	Status	Buy	Sell	Sell Short	Buy to Cover
	Gap_Cohete_Bollin_Erredelugo	Close,20,2,-2	On	On	On		On
	Gap_Submarino_Bollin_Erredelugo	Close,20,2,-2	Off		On	On	On
	Erredelugo_madness_gap	Close,20,2,-2	On	On	On		On

Ilustración 4.8. Esquema portafolio B, sin submarino. Fuente: TradeStation. Elaboración Propia.

De esta forma evitamos pérdidas en las operaciones a corto plazo, obteniendo los siguientes resultados:

Resultados portafolio B: gap Bollinger + madness, sin submarino (a 2 años)			
	ALL trades	LONG trades	SHORT trades
Beneficio bruto	$1.419.659,00	$1.419.659,00	$0,00
Pérdidas brutas	($1.230.318,00)	($1.230.318,00)	$0,00
Beneficio neto	$189.341,00	$189.341,00	$0,00

Tabla 4.9. Resultados portafolio B, sin submarino. Fuente: TradeStation. Elaboración propia.

Sin embargo, los beneficios siguen siendo inferiores a los que hubiésemos obtenidos aplicando únicamente el sistema *madness*. Por tanto, estamos ante la misma situación del portafolio A, donde la mala gestión de efectivo provoca que el algoritmo *Glory Madness* no pueda llegar a su punto máximo debido a que hay capital invertido en operaciones del *gap founder plus*, que son menos rentables.

5. CONCLUSIONES

El desarrollo de un sistema de trading no solo requiere conocimientos sobre programación, matemática, estadística y mercados financieros. Sino que también requiere una base lógica que permita analizar una determinada situación, comprenderla tomando cierta perspectiva y luego influir en ella de cara a alcanzar unos objetivos determinados.

El ejemplo más cercano para representar esta situación es la resolución de un cubo de Rubik. Para llegar a la solución final no solo es necesario conocer de qué color es cada cara, también debemos saber analizar en qué posición nos encontramos, cómo giran las piezas y qué efectos producen la combinación de las mismas.

En este estudio hemos desarrollado cuatro algoritmos de trading para una misma serie, la acción Netflix (NFLX). Cada uno con una forma particular de percibir la realidad e interactuar con ella. Del mismo modo, hemos abarcado dos frecuencias de operativa (diaria e intradiaria) para la misma serie.

La primer conclusión hace referencia al título del presente trabajo, el No Free Lunch Theorem, pues tras los resultados obtenidos debemos afirmar que no puede existir un sistema perfecto de trading que se pueda usar en cualquier circunstancia. Los algoritmos están programados para comprender la realidad mediante parámetros que nosotros mismos definimos, los sistemas simplemente realizan los cálculos que les hemos pedido y actúan en función de estos parámetros. Por tanto debemos programar un algoritmo para cada situación o entorno para que genere unos resultados positivos.

Como segunda conclusión debemos señalar que los modelos cuantitativos pueden variar sustancialmente cuando se aplica sobre una serie temporal. Donde los sucesos aleatorios, los cambios de tendencia y la volatilidad crean situaciones que provocan que los resultados no sean los esperados. Por ello, el papel de la frecuencia con la que nuestro algoritmo lee y actúa en la serie se vuelve esencial a la hora de programar. Pues no es lo mismo programar una estrategia basada en una operativa diaria que en una intradiaria.

Del mismo modo, dos más dos no siempre son cuatro cuando hablamos de sistemas algorítmicos. Ya que al tener una cantidad de recursos limitada, en un tiempo limitado y bajo solo un puñado de opciones posibles, puede fomentar que dos sistemas que funcionan bien por separado den resultados negativos, o menos positivos, si se aplican simultáneamente. De modo que la gestión de capital (money management) es fundamental cuando programamos en un entorno en el que ambas estrategias puedan coexistir.

Finalmente, hemos llegado a la conclusión de que este campo tiene mucho recorrido por delante. Pues día a día la tecnología va en aumento, tenemos más y más información disponible en tiempo real y más capacidad de procesamiento. Por lo que a largo plazo es posible que las finanzas sean más un juego de optimización que de ganadores y perdedores.

6. BIBLIOGRAFÍA

AJRAM, J. (2013): "Bolsa para dummies". Barcelona. Grupo Planeta.

AMAT, O. Y PUIG, X. (2000): "Análisis técnico bursátil". Barcelona. Edicions Gestió 2000.

BERNSTEIN, W. (2008): "A Splendid Exchange: How Trade Shaped the World". Nueva York. Grove Press.

BURNS, A. Y WELLINGS, A. (2003): "Sistemas de tiempo real y lenguajes de programación, 3º edición. Massachusetts. Addison Wesley.

CÁCERES, J. J. (2007): "Conceptos básicos de estadística para ciencias sociales". Delta Publicaciones.

CATALAYUD, F. (2007): "Ponerse corto, abrir corto, cerrar corto" recuperado de: http://www.rankia.com/blog/fernan2/364450-ponerse-corto-abrir-cortos-cerrar

CHICA, V. (2015): " Trading: Technical and fundamental strategies of day trading, invest basics, currency trading and swing trading". Meteor content providers.

FRAGA, U. (2012): "Trading sistemático puro: ventajas e inconvenientes", recuperado de: http://www.novatostradingclub.com/formacion/trading-sistematico-puro-ventajas-e-inconvenientes/

FITSHCHEN, K. (2013): "Building reliable trading systems". Nueva Jersey. John Wiley & Sons.

G. MONTALVO, J. (2014): "Algoritmos y modelos económicos financieros" recuperado de: http://www.econ.upf.edu/~montalvo/noticias/conferencia_bdebate_220514.pdf

HOLLAND, M. (2013): "Issues related to back testing". Recuperado de: http://www.financialtrading.com/issues-related-to-back-testing/

JONES, R. (1999): "The trading game: playing by the numbers to make millions". Jon Wiley & sons.

KOCH, T.W. (2003): "Bank management" . Cincinnati, Ohio. Thomson South-Western.

LITTLE, B. (2013): "A simple gap strategy", recuperado de : http://systemtradersuccess.com/testing-simple-gap-strategy/

PLA V. DE AVILÉS, J. (2013): "Confección de un portafolio de sistemas de trading". Barcelona. Universitat Politècnica de Catalunya. (master Thesis)

TRADESTATION (2015): "EasyLanguage Essentials Programmers Guide" recuperado de: https://www.tradestation.com/~/media/Files/TradeStation/Education/University/School%20of%20EasyLanguage/Books/EL_Essentials.ashx

WOLPERT, D. Y MACREADY, W. (1997): "No free lunch theorems for optimization". IEEE Transactions on Evolutionary Computation, 1(1): 67-82.

WOLPERT, D. Y MACREADY, W. (2005): "Coevolutionary free lunches". IEEE Transactions on evolutionary computation, 9(6): 721-735.