Proyecto Fin de Máster

Máster Universitario en Ingeniería Industrial

Comparativa de algoritmos de aprendizaje para el fraccionamiento óptimo de grandes órdenes bursátiles

Autor: Jair Eddin El Barkani Ismail

Tutor: Daniel Rodríguez Ramírez

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Proyecto Fin de Máster
Máster Universitario en Ingeniería Industrial

Comparativa de algoritmos de aprendizaje para el fraccionamiento óptimo de grandes órdenes

Autor:
Jair Eddin El Barkani Ismail

Tutor:
Daniel Rodríguez Ramírez
Profesor Titular

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2021

Proyecto Fin de Máster: Comparativa de algoritmos de aprendizaje para el fraccionamiento óptimo de grandes órdenes

Autor: Jair Eddin El Barkani Ismail

Tutor: Daniel Rodríguez Ramírez

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2021

El Secretario del Tribunal

Agradecimientos

Gracias a todas esas personas que han estado presente a lo largo de este camino, en especial, a mis padres, mi hermana y Safa, sin ellos esto no habría sido posible.

Gracias a Daniel por la oportunidad brindada.

Jair Eddin El Barkani Ismail

Sevilla, 2021

Resumen

Existen varias técnicas para la división de grandes órdenes bursátiles en pequeñas subórdenes y así reducir su impacto en la bolsa de valores. Estas técnicas requieren de distintas disciplinas como la predicción, redes neuronales, y otros tipos de algoritmos.

En el presente proyecto se han construido dos algoritmos de optimización que utilizan datos locales basados en precios y volúmenes reales de una determinada acción para obtener una secuencia óptima de compra/venta en un horizonte de tiempo reducido.

Ambos surgieron como alternativa al algoritmo [1], en el que se implementan técnicas de predicción. Sin embargo, en este proyecto se prescinde de estrategia de horizonte deslizante y de la optimización en línea reduciéndose todo a un esquema de aprendizaje/estimación. Además, se utilizan técnicas como Local Data llevada a cabo en el algoritmo [2] .

El objetivo del trabajo es plantear un enfoque alternativo para luego compararlos con el algoritmo de partida, y comprobar la viabilidad de estos.

El proyecto se estructura en 5 capítulos. El primero introduce el contexto, y se habla del objetivo y trabajo previo del algoritmo. El segundo abarca el desarrollo del Algoritmo 2. En el tercero se realizan pruebas para comprobar la correcta ejecución de este, y se crea el Algoritmo 3. El cuarto muestra los resultados realizando una comparativa entre ellos. Y, por último, en el quinto capítulo, se redactan las conclusiones.

Abstract

There are several techniques for dividing large stock orders into small sub-orders to reduce their impact on the stock market. These techniques require different disciplines such as prediction, neural networks, and other types of algorithms.

In this project, two optimization algorithms have been built that use local data based on real prices and volumes of a given stock to obtain an optimal purchase / sale sequence in a short time horizon.

Both algorithms emerged as an alternative to Algorithm [1], in which prediction techniques are implemented. However, this project dispenses with the sliding horizon strategy and online optimization, reducing everything to a learning / estimation scheme. Also, techniques such as Local Data carried out in the algorithm [2] are used.

The objective of this project is to propose an alternative approach and then compare it with the starting algorithm to check the viability of the algorithms.

The project is structured in 5 chapters. The first one introduces the context and talks about the objective and previous work of the algorithm. The second one covers the development of the Algorithm 2. In the third one, tests are carried out to verify the correct execution of the algorithm, and Algorithm 3 is developed. The fourth shows the results by comparing the algorithms. And finally, in the fifth chapter, the conclusions are shown.

Índice

Agradecimientos **vii**

Resumen **ix**

Abstract **xi**

Índice **xiii**

Índice de Tablas **xv**

Índice de Figuras **xvii**

1 Introducción **1**
1.1 Trabajo previo y objetivos *2*

2 Descripción del Algoritmo 1 **3**

3 Desarrollo del Algoritmo 2 **7**
3.1. Construcción de la base de datos *7*
3.2. Procedimiento para generar la secuencia *9*
3.2.1 Resolver el problema QP. 10

4 Verificación **13**
4.1 Prueba 1 *13*
4.1.1 Parámetros 13
4.1.2 Resultados prueba 1 14
4.2 Prueba 2 *16*
4.2.1 Parámetros 16
4.2.2 Resultados prueba 2 16
4.3 Conclusiones Prueba 1 y Prueba 2. *18*
4.4 Algoritmo 3 *18*

5 Resultados **21**
4.1 Resultados con horizonte de 120 minutos *21*
5.1.1 Comparativa 21
5.2 Resultados con horizonte de 40 minutos *27*
5.2.1 Comparativa 27

6 Conclusiones **33**

Referencias **35**

Anexo: Códigos usados **37**
Algoritmo 1: Receding Horizon Optimization of Large Trade Orders *37*
Algoritmo 2: Optimización mediante estimación/aprendizaje de la partición de grandes órdenes *42*
Algoritmo 3: Optimización mediante estimación/aprendizaje de la partición de grandes órdenes con ponderación de la Distancia Euclidea *43*
Cargar_muchos_datos_test_generico *45*
DirectWeight *48*

ÍNDICE DE TABLAS

Tabla 1. Algoritmo 1 5
Tabla 2: Base de datos con los estados iniciales y secuencias óptimas asociadas. 8
Tabla 3: Algoritmo 2 9
Tabla 4. Algoritmo 3 19
Tabla 5. Parámetros ajustables caso 1 21
Tabla 6. Precios por sesión caso 1 21
Tabla 7. Precios medios caso 1 22
Tabla 8. Parámetros ajustables caso 2 23
Tabla 9. Precios por sesión caso 2 23
Tabla 10. Precios medios caso 2 24
Tabla 11. Parámetros ajustables caso 3 25
Tabla 12. Precios por sesión caso 3 25
Tabla 13. Precios medio caso 3 26
Tabla 14. Parámetros ajustables caso 4 27
Tabla 15. Precios por sesión caso 4 27
Tabla 16. Precios medios caso 28
Tabla 17. Parámetros ajustables caso 5 29
Tabla 18. Precios por sesión caso 5 29
Tabla 19. Precios medio caso 5 30
Tabla 20. Parámetros ajustables caso 6 31
Tabla 21. Precios por sesión caso 6 31
Tabla 22. Precios medios caso 32

ÍNDICE DE FIGURAS

Figura 1. Esquema del Proyecto 2
Figura 2. Gráfica de precios caso 1 22
Figura 3. Gráfica de precios caso 2 24
Figura 4. Gráfica de precios caso 3 26
Figura 5. Gráfica de precios caso 4 28
Figura 6. Precios por sesión caso 5 30
Figura 7. Precios por sesión caso 6 32

1 INTRODUCCIÓN

La ejecución óptima de órdenes bursátiles, ya sean de compra o venta, en la bolsa de valores supone un reto complicado, así como un problema a la hora de obtener grandes beneficios sin que esto conlleve la asunción de riesgos cuando se invierte en el mercado.

Cuando se trata de grandes órdenes, es inevitable que estas impacten en el mercado en mayor o menor medida, dependiendo obviamente del volumen de compra/venta. Una de las técnicas más utilizadas para reducir este impacto trata de dividir estás grandes órdenes en un conjunto de pequeñas subórdenes que han de ser ejecutadas en un tiempo determinado.

Además, existen básicamente dos tipos de órdenes:

- **Órdenes a mercado**: Son órdenes que se ejecutan sin precio de compra o venta, es decir, que se ejecutan inmediatamente a cualquier precio. Estas son las órdenes en las que se va a enfocar el proyecto.
- **Órdenes limitadas**: Se establece un precio limite, y una vez alcanzado este, se ejecuta la orden. Estas órdenes son más difíciles de dividir en subórdenes, ya que en el tiempo en el que se alcanza el precio límite es posible que no pueda ejecutarse el 100% de las subórdenes.

A la hora de dividir las órdenes en subórdenes existen bastantes maneras de lograrlo; una de las más simples se denomina “Time Weighted Average”, tiempo medio ponderado, TWAP. Este tipo divide la orden en bloques de acciones del mismo tamaño y las ejecuta en intervalos de tiempo regulares.

Por otro lado, otra técnica utilizada, se denomina “Volume Weighted Average Price”, precio medio ponderado por volumen, VWAP. En esta, el tamaño de las subórdenes es proporcional a la predicción del volumen para una determinada fracción de tiempo.

La predicción de los precios de las acciones ha atraído a investigadores de diferentes campos, incluido ingenieros y científicos. Se puede decir, que aún no hay una técnica que se haya aceptado como superior frente a otras; cada una posee sus debilidades y fortalezas.

Cuando se habla de predicción en la bolsa de valores, existen dos tipos de enfoques:

- Por un lado; está el enfoque tradicional financiero, cuyo objetivo es estimar el valor intrínseco de una determinada empresa, que para el caso de una acción podría ser el valor de tiempo descontado del flujo libre de efectivo de una compañía. Este enfoque tiene numerosos problemas de acuerdo con el valor intrínseco, como la necesidad de predecir el flujo de la empresa por numerosos años en el futuro. Otro problema, desde el punto de vista de un inversor es que el valor intrínseco de la empresa y su valor en bolsa en precios, no son lo mismo.
- Por otro lado; otro enfoque a la hora de invertir se define como inversión cuantitativa. En esta estrategia, el objetivo no es estimar el valor intrínseco, sino la tendencia del precio que la acción seguirá en un horizonte de tiempo determinado, que tiende a ser en el corto-medio plazo. La suposición en este tipo de estrategia es que el precio de un valor viene dictado por la fuerza de la oferta y la demanda, y se pueden usar tanto métodos de aprendizaje como estadísticos. Este enfoque asume implícitamente que el precio de una acción no sigue un camino aleatorio, y si fuera aleatorio, significaría que todo el histórico de información de precios sería inútil para intentar predecir los precios de las acciones.

En relación con este último enfoque, existe información limitada en cuanto la ejecución óptima de subórdenes (división de ordenes de mercado) usando técnicas de aprendizaje. Hay más investigaciones sobre predicción de valores mediante diferentes técnicas como:

- Redes neurales en [3].
- Aprendizaje profundo en [4].
- Técnicas basadas en datos locales en [2].
- Máquinas de vectores de soporte en [5].

En el presente proyecto, se pretende realizar una comparativa entre tres algoritmos:

- *Algoritmo 1*: Optimización mediante horizonte deslizante de la partición de grandes órdenes bursátiles [1]. Propone el uso de optimización dinámica sobre un horizonte finito basado en predicciones de precio y volumen para obtener secuencias óptimas de subórdenes, y así cumplir grandes órdenes. Es decir, divide las órdenes en subórdenes de una manera óptima, alcanzando un precio conveniente.
- *Algoritmo 2*: Optimización mediante estimación/aprendizaje de la partición de grandes órdenes bursátiles. Es el que se ha desarrollado en este proyecto, y del que se pretende comprobar su viabilidad, comparándolo con el primero. Este surge de intentar un enfoque alternativo al *Algoritmo 1*. En este se prescinde de la estrategia de horizonte deslizante y de la optimización en línea reduciéndose todo a un esquema de aprendizaje/estimación, ya que en este caso se van a usar datos reales, y no predicciones. Toda la programación se ha realizado en MATLAB R2021a.
- *Algoritmo 3*: Surge tras realizar una serie de ajustes en el Algoritmo 2 con el objetivo de acercar más este a la realidad.

1.1 Trabajo previo y objetivos

El proyecto parte del *Algoritmo 1*. En él se realizan una serie de cambios, para así obtener el *Algoritmo 1 modificado.* Una vez hechas las modificaciones, se puede obtener la base de datos que es el pilar del *Algoritmo 2.* Se desarrolla el *Algoritmo 2,* en el que se obtiene una serie de conclusiones que dan lugar al *Algoritmo 3*. Por último, se realiza una comparación entre los tres, para así ver en cual se obtienen mejores resultados. A continuación, se muestra en la Figura 1, el esquema del proyecto que se ha llevado a cabo.

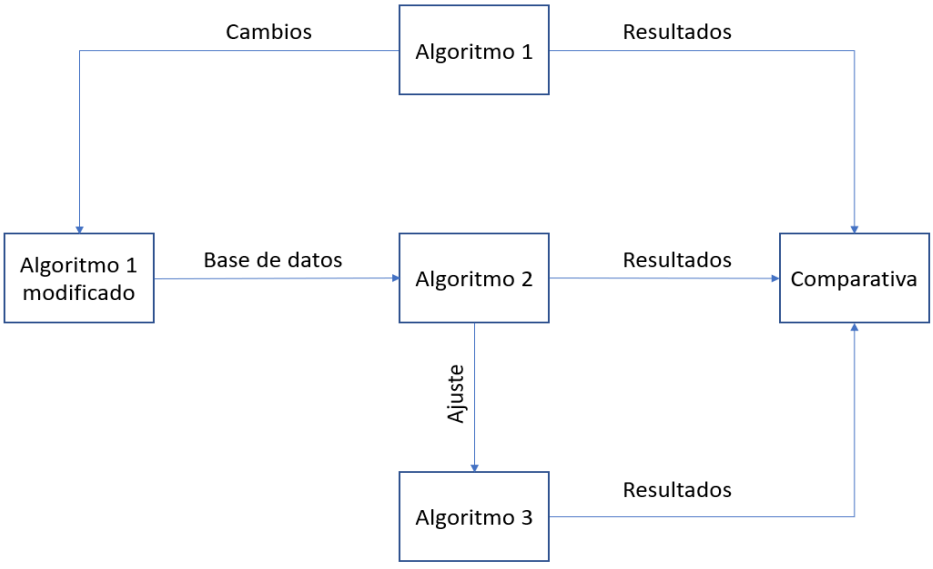

Figura 1. Esquema del Proyecto

En el siguiente apartado se va a describir el Algoritmo 1

2 DESCRIPCIÓN DEL ALGORITMO 1

Este Algoritmo se lleva a cabo en [1], y es el punto de partida de este proyecto.

El objetivo de este es diseñar una estrategia de ejecución de grandes ordenes bursátiles para limitar el impacto en el mercado mediante la división de la orden en un numero pequeño de subórdenes. Es decir, una orden de compra M ϵ $\mathbb{R}^+$de acciones será ejecutada dividiendo la orden en un numero N de subórdenes $m(t+k)$ ϵ $\mathbb{R}^+$, tal que la suma de todas las subórdenes sea igual a M en el caso de que sean órdenes a mercado, si fueran órdenes limitadas tendría que ser el más cercano posible a M.

El número de acciones que son ejecutadas debe ser un numero natural, esto implica que el numero de acciones en las subórdenes que son enviadas al mercado son redondeadas al entero más cercano. Dado que el número de acciones considerado en estas órdenes es bastante grande, a menudo cientos de miles o millones de acciones por orden, la diferencia a nivel financiera entre considerar el número real o entero es despreciable. Aunque desde el punto de vista computacional, la diferencia es bastante significante, siendo mucho más eficiente un algoritmo basado en valores reales de subórdenes.

La estrategia de este algoritmo calcula la división de una orden de forma óptima. Esto es; alcanzando el mejor precio posible. La metodología se basa en la predicción del precio de una acción y del volumen total del mercado en una fracción de tiempo que será definida por N, por ejemplo, desde $t+1$ a $t+N$. El precio y el volumen del mercado pronosticado son usados para calcular un índice de rendimiento que debe ser optimizado. Por lo que la estrategia es un problema de optimización en el que un índice, denominado como V_N, es optimizado.

La formulación de V_N tiene en cuenta ciertos aspectos como el coste total de la orden y el impacto en el mercado de cada suborden, además de las predicciones que son usadas para calcular el índice, también hay un término que penaliza el índice debido a la degradación potencial de las predicciones en un horizonte de predicción.

Sea $\hat{p}(t+k|t)$ el precio pronosticado para el instante $t+k$, $\hat{v}(t+k|t)$ el volumen total pronosticado para el instante $t+k$, y $m(t+k)$ el número de acciones que van a ser compradas en $t+k$, el índice de rendimiento queda definido como:

$$V_N\big(m_N(t),\hat{p}_N(t),\hat{v}_N(t)\big)=\sum_{k=1}^{N}\left(a\hat{p}(t+k|t)+\alpha\left(\frac{m(t+k|t)}{\hat{v}(t+k|t)}\right)^{\beta}\right)m(t+k|t)+\mu\sum_{k=1}^{N}\sigma^{k}\,m(t+k|t) \quad (1)$$

Dónde:

- a = 1 en órdenes de compra y a = -1 órdenes de venta
- α , μ, σ, β son parámetros de ajuste no negativos.
- $m_N(t),\hat{p}_N(t),\hat{v}_N(t)$ son secuencias de subórdenes, precios y volúmenes predichos respectivamente.

Los valores de los parámetros de ajuste que se han usado a la hora de obtener los resultados son:

- $\alpha = 0.1$
- $\beta = 1.1$
- $\sigma = 1.05$
- $\mu = 0.65$

El primer término representa la cantidad pronosticada o la figura económica de comprar o vender una transacción, y además a este termino se le ha incluido un factor de corrección del impacto. Este factor representa la influencia que tiene el volumen de la suborden en el precio para una fracción de tiempo determinada, estos factores son modelados como términos lineales.

El segundo término asigna un mejor coste a las subórdenes que serán ejecutadas en el futuro, ya que el error de predicción crece con el horizonte de predicción

La estrategia propuesta por tanto trata de obtener una secuencia de subórdenes:

$$m_N^*(t) = [m^*(t+1|t), m^*(t+2|t), \dots, m^*(t+N|t)]. \quad (2)$$

Esta secuencia minimiza el índice de rendimiento V_N en una fracción de tiempo. Sin embargo, dependiendo del tipo de orden, algunas constantes deben tenerse en cuenta.

En el caso de las órdenes de mercado, las subórdenes deberían establecer un tamaño límite para que el impacto en el mercado se reduzca. Esto es;

$$0 \leq\ m(t+k|t) \leq\ \overline{m},\ \forall k \in [1, N],$$

Dónde $\overline{m} = 0{,}1M$, que suele ser un valor típico y razonable, es decir, una suborden no puede contener más de un 10% de la suborden. En el caso de una orden limitada el tamaño será el mismo, pero solo si el precio pronosticado alcanza el precio limite p_l. Por ejemplo, si se compra una orden limita $\hat{p}(t+k|t) > p_l$ implica que la compra no puede ser enviada al mercado, es decir, $m(t+k|t) = 0$. Lo mismo ocurre en subórdenes de venta, pero con signo opuesto. Se modifica el tamaño límite:

$$0 \leq\ m(t+k|t) \leq\ \overline{m}_k, \forall k \in [1, N], \quad (3)$$

Donde

$$\overline{m}_k = \begin{cases} 0 & para\ órdenes\ limitadas \text{ cuándo } a(\hat{p}(t+k|t) - p_l) > 0 \\ \overline{m} & en\ caso\ contrario \end{cases} \quad (4)$$

Por otro lado, la suma de todas las subórdenes debería ser igual a el número total de acciones compradas o vendidas, M:

$$\sum_{k=1}^{N} m(t+k|t) = M$$

Esto es más fácil que ocurra en órdenes a mercado, en el caso de órdenes limitadas es posible que no tenga la fracción de tiempo suficiente en la que el precio limite es alcanzado, por lo que a veces la orden no puede ser completamente ejecutada. Para solucionar esto, se incluye una constante que maximiza el grado de cumplimiento de la orden:

$$\sum_{k=1}^{N} m(t+k|t) = M_c \quad (5)$$

Donde:

$$M_c = \min\left\{M, \sum_{k=1}^{N} \overline{m}_k\right\} \quad (6)$$

Este algoritmo utiliza una estrategia de horizonte en retroceso, esto implica que es necesario tener en cuenta que la fracción de tiempo se va reduciendo, por lo tanto, una intervalo de tiempo inicial, N_p tiene que ser considerado. De esta manera inicialmente $N = N_p$, e irá reduciéndose conforme el tiempo avance. Lo mismo ocurre con M debe ir reduciéndose conforme se vayan ejecutando subórdenes, por lo que inicialmente $M = M_I$. Además, esta estrategia de horizonte en retroceso puede ayudar a la hora de mitigar los efectos negativos del error de predicción en el precio para órdenes limitadas, por lo que se vuelve a modificar (4).

$$\overline{m}_k = \begin{cases} 0 & para\ órdenes\ limitadas \text{ cuándo } a(\hat{p}(t+k|t) - p_l) > 0 \ \ y\ k > 1 \\ \overline{m} & en\ caso\ contrario \end{cases} \tag{7}$$

Una vez definidas todas las constantes necesarias, la secuencia de subórdenes se obtiene resolviendo el siguiente problema de optimización:

$$m_N^*(t) = \min_{m_N(t)} V_N(m_N(t), \hat{p}_N(t), \hat{v}_N(t)) \tag{8}$$

$$st \quad 0 \leq m(t+k|t) \leq \overline{m}_k, \forall k \in [1, N], \\ \sum_{k=1}^{N} m(t+k|t) = M_c$$

En la Tabla 1, se puede observar la estrategia del Algoritmo 1 a modo de resumen.

Algoritmo 1. Optimización mediante horizonte deslizante de la partición de grandes órdenes

Requiere: $M_I, N_p, a, \alpha, \beta, \mu, \sigma, \overline{m}$.

1. $M \leftarrow M_I$.
2. $N \leftarrow M_p$.
3. **Repetir**
4. Calcula las predicciones de $\hat{p}_N(t)\ y\ \hat{v}_N(t)$.
5. Calcula los límites de $\overline{m}_k$ como en (7)
6. Calcula M_c como en (6)
7. Resuelve (8) para obtener $m_N^*(t)$
8. Espera al siguiente tiempo, que es, $t + 1$
9. Calcula:

$$m_{t+1}^* = \begin{cases} 0 & para\ órdenes\ limitadas \text{ cuándo } a(\hat{p}(t+k|t) \quad - p_l) > 0 \\ m^*(t+1|1) & en\ caso\ contrario \end{cases}$$

10. **if** $m_{t+1}^* > 0$ **then**
11. Se envía m_{t+1}^* al mercado
12. $M \leftarrow M - m_{t+1}^*$
13. **end if**
14. $N \leftarrow N - 1$
15. **until** $N = 0\ or\ M = 0$

Tabla 1. Algoritmo 1

Los resultados que se aportaron en [1], fueron bastante positivos a pesar del hecho de que el horizonte de predicción es bastante largo. La cantidad potencial de ahorro ante las estrategias conocidas del mercado, han sido bastante significativa.

En el presente proyecto, solo se han utilizado órdenes a mercado de compra. A la hora de construir el Algoritmo 2, que se desarrolla en el apartado siguiente, se lleva a cabo bajo esta premisa, por lo que a la hora de compararlos solo se hará para órdenes a mercado de compra.

3 DESARROLLO DEL ALGORITMO 2

Como se ha comentado anteriormente, este algoritmo surge de intentar un enfoque alternativo al Algoritmo 1, en el que se proponía una optimización mediante horizonte deslizante de la partición de grandes órdenes bursátiles.

En éste, cada orden, que inicialmente serán de 2.000.000 de acciones, se debe distribuir en un intervalo de 120 minutos, que es la duración de una sesión en el mercado asiático, en el que se van a realizar las pruebas.

La operación parte de un estado inicial $Z_I(t)$ el cuál se calculará usando datos reales (no predicciones) de precios y volúmenes futuros, a diferencia del Algoritmo 1. La optimización da como lugar una secuencia de 120 órdenes, 120 decisiones de compra/venta.

Por tanto, lo que se va a realizar es lo siguiente:

1. Construcción de un base datos en la que a cada estado inicial se le asocie la secuencia óptima de 120 órdenes.
2. La base de datos anterior se usará después para calcular la partición óptima de una orden de 2.000.000 de acciones.
3. Procedimiento para generar la secuencia óptima, mediante un problema de QP.
4. Una vez calculada la secuencia óptima se usarán los valores de la secuencia en cada instante del intervalo de la sesión.

3.1. Construcción de la base de datos

Para construir la base de datos BD, se va a necesitar realizar modificaciones en el Algoritmo 1.

Las modificaciones son las siguientes:

- Eliminar las predicciones; solo se usan datos reales, tanto de volumen como de precios.
- Eliminar términos en el índice de rendimiento relacionados con las predicciones, y los errores en el horizonte deslizante, en concreto el parámetro de ajuste μ tiene que ser 0, el resto de parámetros de ajuste permanecen igual.
- Añadir la expresión dónde se almacena la base de datos, BD.

En primer lugar, el Algoritmo 1, carga un conjunto de datos, conjunto de validación, en el que contienen los datos de precios y volúmenes relacionados con una acción.

Los conjuntos de validación se han obtenido mediante la función Cargar_muchos_datos.m, que carga un Excel en el que se encuentran los datos de los precios y volúmenes por minuto de una determinada acción; en el caso de este proyecto se han usado los datos de la empresa, Ping An Bank, entre el 1/4/2017 y el 12/29/2020.

Estos conjunto de datos son almacenados en distintas variables. Las variables que se necesitan son las relacionadas con los precios y volúmenes reales.

Por ejemplo, las variables relacionadas con los precios y los volúmenes serían:

- X_Price_test, dimensión: (sesión, tamaño regresor)

  - dónde la sesión corresponde a una sesión de 120 minutos,

  - y el tamaño del regresor corresponde a los minutos de la sesión que se quieran utilizar.

- X_Volume_test: (sesión, tamaño regresor)

  - dónde la sesión corresponde a una sesión de 120 minutos,

  - y el tamaño del regresor corresponde a los minutos de la sesión que se quieran utilizar.

El estado inicial $Z_I(t)$, estará formado por la variable X_Price_test, que contiene los precios de una determinada sesión, y la última componente del vector sería X_Volume_test, indicando el volumen medio de esa sesión:

$$Z_I(t) = [\, X\ Price\ test\ (:, tam\ regresor), X\ volume\ test(:,1)\ ]$$

La base de datos estará formada por estados iniciales con su correspondiente secuencia óptima de 120 órdenes, (decisiones de compra/venta). Es decir, en cada línea de la base datos aparece la condición inicial que se dio en el pasado y su secuencia óptima calculada antes de empezar la sesión; esto es:

$$\boldsymbol{m_i^*(t)} = [m_i^*(t+1), \dots, m_i^*(t+120)]$$

$Z_I^1(t_1)$	$\boldsymbol{m_1^*}(t_1)$
$Z_I^2(t_2)$	$\boldsymbol{m_2^*}(t_2)$
$Z_I^3(t_3)$	$\boldsymbol{m_3^*}(t_3)$
.	.
.	.
.	.
$Z_I^L(t_L)$	$\boldsymbol{m_L^*}(t_L)$

Tabla 2: Base de datos con los estados iniciales y secuencias óptimas asociadas.

La Tabla 2, contiene los estados iniciales y secuencias óptimas asociadas de L sesiones de bolsa; cada una comienza en $t_i + 1$ y con 120 subórdenes, por lo que $\boldsymbol{m_i^*}(t_i) \in \boldsymbol{R}^{120}$.

Esta base de datos se usará para calcular la partición óptima de una orden de 2.000.000 de acciones. Así, antes de empezar el intervalo; se formará el vector inicial $Z_I(t)$ y a partir de ese estado inicial y de la base de datos BD, se generará una secuencia óptima $\widehat{\boldsymbol{m}}^*(\boldsymbol{t})$, la cual se aplicará a razón de una suborden cada minuto. Es decir, la estimación de la secuencia óptima solo se calcula una vez, antes de comenzar la sesión, y todas las subórdenes de la secuencia estimada se enviarán al mercado en el instante que le corresponda.

3.2. Procedimiento para generar la secuencia

Para generar la secuencia óptima, se necesitan como datos de entrada:

- La base de datos, DB.
- El estado inicial, $Z_I(t)$
- El número de entradas más cercanas o parecidas al estado inicial, N.

La Tabla 2, muestra el procedimiento de estimación de la secuencia óptima.

Algoritmo 2. Optimización mediante estimación/aprendizaje de la partición de grandes órdenes

Input: DB, $Z_I(t), N$.

Output: $\widehat{\boldsymbol{m}}^*(\boldsymbol{t})$ (estimación de la secuencia óptima en t)

1. Calcular la distancia euclídea d $(Z_I(t), Z_I^i(t_i))$, para todos los $Z_I^i(t_i)$ de la base de datos DB.
2. Crear una lista de las entradas de la base de datos DB ordenada según su distancia d $(Z_I(t), Z_I^i(t_i))$.
3. Construir un conjunto de $\Omega(Z_I(t)))$ usando las primeras N entradas de la lista ordenada (es decir, usando las N entradas más cercanas o parecidas).
4. Resolver el siguiente problema QP:

$$[\lambda_1^*, \lambda_2^*, \dots, \lambda_N^*] = \arg\ \min \sum_{i=1}^N \lambda_i^2 \quad (1)$$

$$\text{St} \quad \sum_{i=1}^N \lambda_i = 1, \quad (2)$$

$$\sum_{i=1}^N \lambda_i Z_I^i(t_i) = Z_I(t), \quad (3)$$

5. Calcular: $\widehat{\boldsymbol{m}}^*(\boldsymbol{t})$ como:

$$\widehat{\boldsymbol{m}}^*(\boldsymbol{t}) = \sum_{i=1}^N \lambda_i\, m_i^*(t)$$

Tabla 3: Algoritmo 2

La base de datos, DB, contiene numerosos estados o sesiones de mercado pasados, en concreto 1500 sesiones, cada una de ellas formada por 120 minutos. Cada sesión a su vez tiene asignado un vector secuencia óptima de compra. Por tanto, en relación con la Tabla 1, la matriz tendrá de dimensión (1500x221):

- 1500 correspondientes al número de sesiones,
- y 221 se divide en: 100 correspondiente a los minutos (aunque en realidad son 120 minutos cada sesión, se ha establecido 100 minutos como tamaño máximo de regresor), 1 correspondiente al volumen por minuto, y luego por otro lado 120 subórdenes (decisiones de compra/venta), de acuerdo con esos 120 minutos.

El estado inicial $Z_I(t)$ es el que va a ser comparado con los estados que contiene la base de datos, para comprobar cuál de ellos se parece más a $Z_I(t)$. Para ello, se calcula la distancia euclídea entre los vectores, para posteriormente ordenarlos de mayor a menor proximidad.

N, es el número de vectores de esa lista ordenada que se van a usar para calcular la secuencia óptima como combinación de N estados.

Por tanto, el objetivo del algoritmo es encontrar una combinación de N estados lo más próximo posible al estado inicial $Z_I(t)$, y ponderar la secuencia óptima de cada uno de esos estados, para obtener la secuencia óptima para el instante siguiente.

3.1.1 Resolver el problema QP.

Como se observa en la Tabla 2, se ha de resolver un problema cuadrático. Para ello se va a hacer uso de la función Quadprog, del paquete de MATLAB Optimization toolbox.

Esta función de MATLAB, obedece a la siguiente función de optimización:

$$\begin{aligned} \min \quad & \frac{1}{2}\, x^T H\, x + \, f^T x \\ st \quad & A\, x \, \leq b, \\ & Aeq\; x = beq \\ & lb \, \leq x \leq ub \end{aligned}$$

La sintaxis de quadprog es la siguiente:

[x,fval,exitflag,output] = quadprog(H, f, A, b, Aeq, beq, lb , ub, x0, options)

Donde:

- X es la incógnita del problema; en este proyecto, corresponde a la variable lambda, λ, la cual se utiliza para combinar el numero N de estados de la base de datos, para obtener el inicial.
- Fval, devuelve el valor de la solución de la función de coste.
- Exitflag, devuelve valor que describe la condición de salida, en este caso si es 1, significa que se ha resuelto satisfactoriamente.
- Output, devuelve una estructura con información sobre el proceso de optimización.
- H y f son valores de la función de optimización.
- A y b son parámetros sujetos a condición de desigualdad.
- Aeq y beq son parámetros sujetos a condición de igualdad.
- lb y ub, son los límites inferior y superior para la variable.

Si se compara esta función de optimización con el problema del Algoritmo 2, es evidente que hay que realizar una serie de ajustes para asemejar el problema; entre ellos:

- Ajustar la función objetivo:, donde λ= x, hay que dar valores a H y f tal que la función objetivo quede λ_i^2, esto es:
    - H = 2 eye (N) , una matriz identidad multiplicada por 2 para eliminar el $\frac{1}{2}$, y que el resultado de multiplicar X por su traspuesta de x^2
    - f = zeros(N,1), un vector de ceros para que al multiplicar por x^T de como resultado 0.

- Añadir restricciones de desigualdad:
    - A = [-BDordenada(1:N,102:221)'; BDordenada(1:N,102:221)']
    - b = [zeros(120,1) ; 2000*ones(120,1)]

    Estos parámetros corresponden a dos restricciones: la primera es que todas las subórdenes de compra han de ser mayores que 0, y la segunda que cada secuencia de subórdenes de compra han de ser igual a 2.000.000 de acciones.

- Añadir restricciones de igualdad:

    - Aeq = [BDordenada(1:N,1:tam_regresor)'; ones(1,N)];
    - Beq = [X_Price_test(k,1:tam_regresor)' ; 1];

    Estos parámetros corresponden también a dos restricciones: la primera es que la combinación de estados de la base de datos ponderada por λ_i ha de ser igual al estado inicial $Z_I(t)$, y la segunda es que la sumatoria de todos los λ_i ha de ser igual a 1.

Una vez definidos todos los parámetros se resuelve el problema y se obtiene la secuencia óptima como:

$$\widehat{\boldsymbol{m}}^*(\boldsymbol{t}) = \sum_{i=1}^{N} \lambda_i \, m_i^*(t)$$

O lo que es lo mismo:

$\widehat{\boldsymbol{m}}^*(\boldsymbol{t})$ = transpose(lambda' * BDordenada(1:N,102:221))

Tras la obtención de la secuencia óptima se pueden obtener los precios medios a los que se han comprado las acciones. Se multiplica los precios en el instante siguiente por la secuencia óptima y luego se divide entre 2.000.000 de acciones.

Precio (n) = (Y_Price_real_test_h (1,:,n) * m)/ 20000;

4 VERIFICACIÓN

Una vez terminado el algoritmo, se debe comprobar que está bien construido para así corroborar la viabilidad de este.

Para comprobar el algoritmo, se va a realizar lo siguiente:

1. Se va a crear una base de datos que contiene los estados pasados de los precios de la empresa china Ping An Bank, cuyo ticker es *000001*. Esta empresa posee una gran capitalización.
    a. Para crear la base de datos, se va a utilizar el Algoritmo 1 modificado como bien se ha explicado con anterioridad, cargando en él, el conjunto de validación relacionado con dicha empresa. Este algoritmo creará la base de datos con 1500 sesiones de 120 minutos, y a cada sesión le asignará sus 120 subórdenes de compra, cuya suma debe dar 2.000.000 de acciones.
2. Una vez creada la base de datos, se utiliza el Algoritmo 2, cuyos input son: La base de datos, BD, el estado inicial, $Z_I(t)$, y el numero de entradas o lambdas de la lista ordenada que más se parecen al estado inicial, N.

4.1 Prueba 1

En esta prueba, lo que se pretende es evaluar un estado inicial, que esté contenido en la base de datos. Si esto fuera así, la combinación lineal de los estados solo tendría una componente, que sería $\lambda_1 = 1$, y el resto de lambdas serían 0, ya que si se multiplica este lambda por el estado de la base de datos que es igual al estado inicial, se obtendría el estado inicial, y, por consiguiente, la secuencia óptima asociada.

4.1.1 Parámetros

Los parámetros utilizados del Algoritmo 2 son:

- N = 20, lo que significa 20 estados más parecidos; implica 20 posibles lambdas que van a ser combinadas para obtener el estado inicial.
- Tam_regresor = 100, se va a utilizar 100 componentes de los vectores estados, correspondiente a los 100 minutos de la sesión.
- n = 1, 2.. es la sesión que se va a comprobar del estado inicial.

Al ejecutar el algoritmo, en primer lugar, se define el estado inicial bajo la variable:

X_Price_test = X_Price_test_h (1,tam_regresor,n)

Luego, este vector se compara con la base de datos, y se crea una lista de N estados más parecidos al estado inicial, esto es, el paso 2 del algoritmo 2, en el que se calcula la distancia euclídea.

Cuando ya se tiene la lista ordenada, se resuelve el problema de QP, y es aquí dónde se comprueba el correcto funcionamiento del algoritmo.

4.1.2 Resultados prueba 1

- Para la sesión, n = 1: Todos los lambdas son 0 excepto la primera componente.

$$\lambda = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Esto significa que, en la lista de la base de datos de 20 estados, el primer estado es idéntico al estado inicial. Por lo que su secuencia óptima de subórdenes de compras será idéntica, y solo será combinación de ella misma, es decir:

$$Z_I = \lambda' . BDordenada(1{:}100)$$

Y el vector asociado a las subórdenes de compra será:

$$\widehat{\boldsymbol{m}}^* = \lambda' . BDordenada(100{:}221)$$

Este vector de subórdenes de compra, si se sumaran todas sus componentes, sería igual a 2.000.000 acciones.

Si quisiéramos comprar con esta secuencia, tendríamos que multiplicar este vector por los precios futuros:

$$\text{Y\_Price\_real\_test\_h}(1,:,n) * \widehat{\boldsymbol{m}}^*$$

Si dividimos este valor por el número de acciones compradas, obtenemos el precio medio de compra de las acciones:

$$Precio\ Medio = \frac{\text{Y\_Price\_real\_test\_h}(1,:,n) * \widehat{\boldsymbol{m}}^*}{2M} = 12.7840\ u.m$$

- Para la sesión, n = 2: Ocurriría lo mismo, $\lambda_1 = 1$, y el resto serían 0. Es así porque se vuelve a ordenar la lista y el primer vector es el más parecido; en este caso idéntico al estado inicial de la sesión n = 2.

$$\lambda = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

La secuencia óptima depende únicamente del primer estado, pues será la asociada en el Algoritmo 1.

El precio medio en este caso sería:

$$Precio\ Medio = \frac{\text{Y\_Price\_real\_test\_h}(1, :, \text{n}) \ * \ \widehat{\boldsymbol{m}}^*}{2M} = 10.7107\ u.m$$

- Para la sesión, n = 3: Se encuentra un escenario diferente. En primera instancia, se pensó que pudo ser un fallo del algoritmo, ya que no se obtuvo $\lambda_1 = 1$, que es lo que había estado ocurriendo en los demás casos, al encontrarse el estado inicial contenido en la base de datos.

  Lo que ocurría es que el estado n = 3, estaba contenido 2 veces en la base de datos, y esto daba como solución, $\lambda_1 = 0.5$, $\lambda_2 = 0.5$, pues es totalmente lógico, al haber 2 estados equidistantes en la base de datos al estado inicial, se usa la combinación lineal de los 2.

$$\lambda = \begin{pmatrix} 0.5 \\ 0.5 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Como son los dos estados iguales, y también tienen la misma secuencia de subórdenes asociada, daría lo mismo que fuese $\lambda_1 = 1, \lambda_2 = 0,$ o cualquier combinación cuya sumatoria entre estas dos componentes sea 1.

El precio medio en este caso:

$$Precio\ Medio = \frac{\text{Y\_Price\_real\_test\_h}(1, :, \text{n}) \ * \ \widehat{\boldsymbol{m}}^*}{2M} = 16.0600\ u.m$$

4.2 Prueba 2

En esta prueba, se va a realizar lo mismo que en la prueba 1, pero modificando los parámetros del algoritmo. Se va a comprobar la consistencia aumentando el número de entradas de la lista ordenada, es decir, utilizar más lambdas a la hora de obtener la combinación del estado inicial.

4.2.1 Parámetros

- N = 200, lo que significa 200 estados más parecidos, implica 20 posibles lambdas que van a ser combinadas para obtener el estado inicial.
- Tam_regresor = 100, se va a utilizar 100 componentes de los vectores estados, correspondiente a los 100 minutos de la sesión.
- n = 1, 2.. es la sesión que se va a comprobar del estado inicial.

4.2.2 Resultados prueba 2

- Para la sesión n = 1, cuándo se esperaba obtener un $\lambda_1 = 1$, y el resto 0, como anteriormente, al haber tantos lambdas, el solucionador encuentra otras soluciones óptimas. Utiliza casi todos los lambdas para obtener la combinación del estado inicial. Es decir, utiliza 200 estados de la base de datos, combinándolos para obtener el estado inicial. Esto es:

$$\lambda = \begin{pmatrix} 0.5137 \\ 0.0307 \\ 0.0193 \\ 0.0193 \\ 0.0095 \\ -0.0045 \\ \vdots \\ -0.0369 \end{pmatrix}$$

$$Z_I = \lambda' . BDordenada(1{:}100)$$
$$\widehat{\boldsymbol{m}}^* = \lambda' . BDordenada(100{:}221)$$

Ahora la secuencia de subórdenes óptima será la combinación de todas las secuencias de subórdenes de los 200 estados.

El precio medio que se obtendría con esta secuencia será:

$$Precio\ Medio = \frac{\text{Y\_Price\_real\_test\_h}(1,:,\text{n}) \ * \ \widehat{\boldsymbol{m}}^*}{2M} = \ 12.8173\ u.m$$

Si lo comparamos con el precio obtenido en la Prueba 1, el precio medio ha aumentado:

$$\%Aumento = \frac{12.8173 - \ 12.7840}{12.7840}\ x\ 100 = 0.26\ \%$$

El precio ha aumentado un 0,26%, luego se puede concluir en este caso que, a pesar de ser una cantidad pequeña, se ha obtenido un resultado peor, ya que el precio medio de compra es mayor.

- Para la sesión n = 2 se tiene:

$$\lambda = \begin{pmatrix} 0.5500 \\ -0.0009 \\ -0.0107 \\ 0.0222 \\ 0.0068 \\ -0.0121 \\ \vdots \\ -0.0051 \end{pmatrix}$$

En este caso, el precio medio de compra ha sido:

$$Precio\ Medio = \frac{\text{Y\_Price\_real\_test\_h}(1,:,\text{n}) \ * \ \widehat{\boldsymbol{m}}^*}{2M} = 10.7802\ u.m$$

Luego si lo comparamos con el Precio medio obtenido en la Prueba 1, el precio también ha aumentado.

$$\%Aumento = \frac{10.7802 - \ 10.7107}{10.7107}\ x\ 100 = 0.65\ \%$$

El precio ha aumentado un 0.65%, luego igual que anteriormente, el resultado es peor.

4.3 Conclusiones Prueba 1 y Prueba 2.

Al contrastar los resultados, es evidente que estos son mejores en la prueba 1, ya que se ha obtenido un precio menor de compra. Esto es debido a que, al tener un número reducido de estados, el solucionador encuentra la solución que es igual que el estado inicial, con solo un estado de la base de datos, a diferencia de la prueba 2, que utiliza todos los estados (N=200), para combinarlos y obtener el estado inicial, independientemente de si son más cercanos o no, los utiliza todos; es decir, encuentra una función de coste menor, cuándo combina los 200 estados.

Pero ¿por qué son mejores?.

Pues, en la base de datos, cada estado tiene asociado su secuencia óptima de subórdenes, entonces al tener un estado inicial que es igual que al menos uno de los estados de la base, su secuencia óptima ya está calculada, y es difícil, encontrar una combinación de estados en las que se obtenga una secuencia óptima mejor que la presente en la base.

Esto solo ocurre porque estamos utilizando estados que están contenidos en la base.

Además, cuándo se utilizan muchos estados de la base de datos, N=200 o más, el algoritmo busca la solución que es igual al estado inicial, pero sin tener en cuenta cuales son los estados más cercanos al estado inicial. Esto nos da soluciones que no se parecen a la realidad.

Es por esto, que surge la necesidad de realizar un ajuste al algoritmo, para que este tenga en cuenta aquellos estados más cercanos al estado inicial cuándo se está usando una numerosa cantidad de lambdas o entradas.

Para ello se va a modificar la función de coste. Se le va a incluir la distancia euclídea entre los estados de la base de datos, y un parámetro γ de ponderación. Al insertar la distancia euclídea, como es una función objetivo de minimización, el algoritmo intentará combinar con mayor proporción aquellos estados más cercanos al estado inicial.

4.4 Algoritmo 3

Tras la necesidad de modificar el Algoritmo 2, surge la implementación de un nuevo Algoritmo, que también va a ser comparado en los resultados del proyecto.

A modo de resumen, en la Tabla 4, se observa el Algoritmo 3. Optimización mediante estimación/aprendizaje de la partición de grandes ordenes con ponderación de la Distancia Euclídea**.**

Algoritmo 3. Optimización mediante estimación/aprendizaje de la partición de grandes ordenes con ponderación de la Distancia Euclídea.

Input: DB, $Z_I(t), N$.

Output: $\widehat{\boldsymbol{m}}^*(\boldsymbol{t})$ (estimación de la secuencia óptima en t)

16. Calcular la distancia euclídea d ($Z_I(t)$, $Z_I^i(t_i)$), para todos los $Z_I^i(t_i)$ de la base de datos DB.
17. Crear una lista de las entradas de la base de datos DB ordenada según su distancia d ($Z_I(t)$, $Z_I^i(t_i)$).
18. Construir un conjunto de $\Omega(Z_I(t))$ usando las primeras N entradas de la lista ordenada (es decir, usando las N entradas más cercanas o parecidas).
19. Resolver el siguiente problema QP:

$$[\lambda_1^*, \lambda_2^*, \dots, \lambda_N^*] = \arg \min \sum_{i=1}^N \lambda_i^2 \, (\gamma + d_i) \quad (1)$$

$$\text{St} \quad \sum_{i=1}^N \lambda_i = 1, \quad (2)$$

$$\sum_{i=1}^N \lambda_i Z_I^i(t_i) = Z_I(t), \quad (3)$$

20. Calcular: $\widehat{\boldsymbol{m}}^*(\boldsymbol{t})$ como:

$$\widehat{\boldsymbol{m}}^*(\boldsymbol{t}) = \sum_{i=1}^N \lambda_i \, m_i^*(t)$$

Tabla 4. Algoritmo 3

En el código, la modificación se hace en la matriz H, que está contenida en la función de coste:

```
H = 2*(diag(BD2(1:N,1)) + gamma*eye(N))
```

En el siguiente apartado, se van a calcular secuencias óptimas para estados que no están contenidos en la base, y comprobaremos con qué número de estados se obtienen mejores resultados. Es decir, si se obtienen mejores resultados con mayor combinación de lambdas o menores.

5 RESULTADOS

4.1 Resultados con horizonte de 120 minutos

Los resultados han sido clasificados en base a los parámetros ajustables del algoritmo. Estos son: el tamaño del regresor, el número N de entradas más parecidas al estado inicial y el numero de sesiones. Además, para el Algoritmo 3, se obtendrán diferentes resultados en función del parámetro de ponderación γ.

El objetivo es obtener el menor precio medio de compra posible. Se va a probar con distintos ajustes para ver en cual se obtiene mejor precio.

A continuación, se muestran diferentes casuísticas en las que se van a comparar los tres algoritmos.

5.1.1 Comparativa

- CASO 1:
  - Parámetros ajustables:

	Parámetros		
	tam_regresor	N entradas	n_test sesiones
Algoritmo 1	50	-	20
Algoritmo 2	50	200	20
Algoritmo 3	50	200	20

Tabla 5. Parámetros ajustables caso 1

  - Precios por sesión:

		Sesión									
		1	2	3	4	5	6	7	8	9	10
	Algoritmo 1	13,1428	13,9170	13,0057	13,4590	14,1621	14,3380	12,7063	14,5001	13,1040	14,5800
	Algoritmo 2	13,1436	14,0044	13,0048	13,4604	14,1501	14,4224	12,7140	14,5006	13,0403	15,0335
γ = 0.1	Algoritmo 3	13,1662	13,9387	13,0230	13,4650	14,1305	14,4428	12,7152	14,5060	13,0575	15,0462
γ = 0.01	Algoritmo 3	13,1656	13,9357	13,0210	13,4639	14,1321	14,4410	12,7148	14,5061	13,0566	15,0447
γ = 0.001	Algoritmo 3	13,1654	13,9355	13,0206	13,4637	14,1323	14,4408	12,7147	14,5061	13,0564	15,0448

		Sesión									
		11	12	13	14	15	16	17	18	19	20
	Algoritmo 1	14,1439	12,6490	13,4743	15,2050	12,3510	13,3155	13,5885	14,7850	13,0316	13,4251
	Algoritmo 2	14,1027	12,7006	13,5111	15,3553	12,4436	13,3810	13,6160	14,8784	13,0349	13,4637
γ = 0.1	Algoritmo 3	14,0996	12,6717	13,5147	15,3538	12,3789	13,3669	13,6075	14,8669	13,0188	13,4203
γ = 0.01	Algoritmo 3	14,0992	12,6715	13,5149	15,3537	12,3777	13,3668	13,6076	14,8669	13,0194	13,4175
γ = 0.001	Algoritmo 3	14,0992	12,6715	13,5149	15,3537	12,3778	13,3667	13,6075	14,8669	13,0195	13,4171

Tabla 6. Precios por sesión caso 1

Observando la tabla de precios, dependiendo de la sesión, se obtienen mejores precios en un algoritmo o en otro. En la Figura 2, se puede observar de manera gráfica en qué algoritmo se obtiene menores precios por sesión.

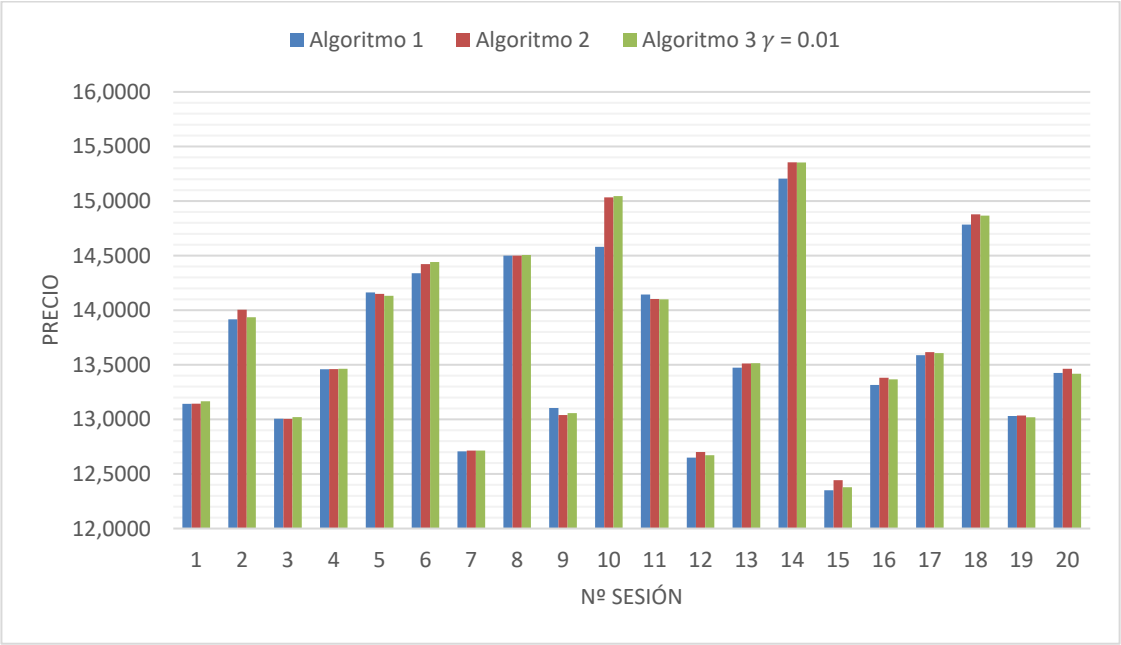

Figura 2. Gráfica de precios caso 1

En la Tabla 7, se puede observar que se ha obtenido mejor precio de compra en el Algoritmo 1, ya que el precio medio de compra es menor.

	Precio medio Algoritmo 1	13,6442
	Precio medio Algoritmo 2	13,6920
γ = 0.1	Precio medio Algoritmo 3	13,6895
γ = 0.01	Precio medio Algoritmo 3	13,6888
γ = 0.001	Precio medio Algoritmo 3	13,6888

Tabla 7. Precios medios caso 1

- CASO 2:
  - Parámetros ajustables: En comparación con el Caso 1, se ha aumentado el número de entradas para comprobar su impacto en el precio. Al haber mayor número de entradas, habrá una combinación mayor de estados para conseguir el estado inicial.

	Parámetros		
	tam_regresor	N entradas	n_test sesiones
Algoritmo 1	50	-	20
Algoritmo 2	50	250	20
Algoritmo 3	50	250	20

Tabla 8. Parámetros ajustables caso 2

  - Precios por sesión: En la Tabla 9, se puede observar un ligero cambio en los precios del Algoritmo 2. Se han reducido en algunos casos.

		Sesión									
		1	2	3	4	5	6	7	8	9	10
	Algoritmo 1	13,1428	13,9170	13,0057	13,4590	14,1621	14,3380	12,7063	14,5001	13,1040	14,5800
	Algoritmo 2	13,1374	14,0277	13,0049	13,4605	14,1554	14,4391	12,7159	14,4987	13,0481	15,0022
$\gamma = 0.1$	Algoritmo 3	13,1638	13,9472	13,0132	13,4649	14,1495	14,4487	12,7108	14,5030	13,0519	15,0390
$\gamma = 0.01$	Algoritmo 3	13,1644	13,9433	13,0121	13,4639	14,1507	14,4469	12,7110	14,5035	13,0521	15,0320
$\gamma = 0.001$	Algoritmo 3	13,1642	13,9427	13,0119	13,4637	14,1509	14,4467	12,7110	14,5036	13,0521	15,0311

		Sesión									
		11	12	13	14	15	16	17	18	19	20
	Algoritmo 1	14,1439	12,6490	13,4743	15,2050	12,3510	13,3155	13,5885	14,7850	13,0316	13,4251
	Algoritmo 2	14,1056	12,6843	13,5150	15,3525	12,4230	13,3642	13,6142	14,8506	13,0251	13,4607
$\gamma = 0.1$	Algoritmo 3	14,0998	12,6807	13,5146	15,3531	12,3517	13,3636	13,6030	14,8607	13,0197	13,4351
$\gamma = 0.01$	Algoritmo 3	14,1004	12,6802	13,5154	15,3529	12,3547	13,3652	13,6032	14,8607	13,0195	13,4288
$\gamma = 0.001$	Algoritmo 3	14,1004	12,6801	13,5156	15,3529	12,3552	13,3655	13,6033	14,8607	13,0195	13,4277

Tabla 9. Precios por sesión caso 2

En la Figura 3, se puede observar de manera gráfica en que algoritmo se obtiene menores precios por sesión. Se elige el mejor precio del Algoritmo 3 a la hora de representarlo en la gráfica.

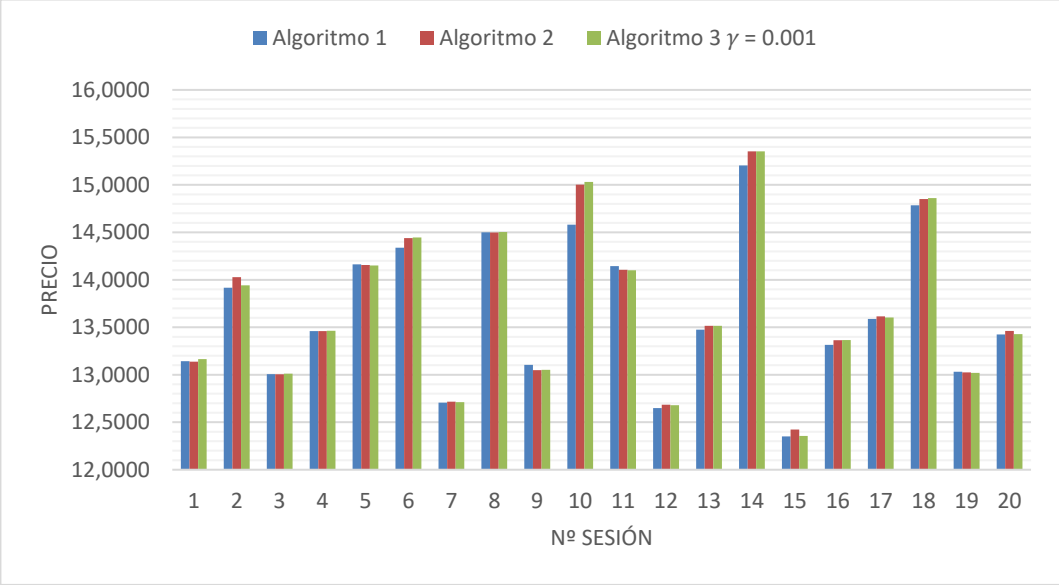

Figura 3. Gráfica de precios caso 2

En este caso, se ha obtenido una ligera bajada de precios en el Algoritmo 2, pero sigue sin ser suficiente. El Algoritmo 1 obtiene mejores precios.

	Precio medio Algoritmo 1	13,6442
	Precio medio Algoritmo 2	13,6899
γ = 0.1	Precio medio Algoritmo 3	13,6887
γ = 0.01	Precio medio Algoritmo 3	13,6880
γ = 0.001	Precio medio Algoritmo 3	13,6879

Tabla 10. Precios medios caso 2

- CASO 3:
  - Parámetros ajustables: En este caso se han modificado la entrada y el tamaño del regresor, es decir, ahora se utilizará todas las componentes de los vectores estado.

	Parámetros		
	tam_regresor	N entradas	n_test sesiones
Algoritmo 1	100	-	20
Algoritmo 2	100	300	20
Algoritmo 3	100	300	20

Tabla 11. Parámetros ajustables caso 3

  - Precios por sesión: En este caso han modificado los precios los 3 algoritmos, debido al incremento de componentes.

		Sesión									
		1	2	3	4	5	6	7	8	9	10
	Algoritmo 1	13,1751	14,0006	12,9566	13,4521	14,1830	14,4261	12,7169	14,4893	13,0486	14,5800
	Algoritmo 2	13,1342	13,9877	13,0039	13,4582	14,1485	14,4375	12,7158	14,5079	13,0495	15,0628
$\gamma = 0.1$	Algoritmo 3	13,1598	13,9704	12,9931	13,4594	14,1232	14,4477	12,7117	14,5035	13,0381	15,0187
$\gamma = 0.01$	Algoritmo 3	13,1599	13,9712	12,9908	13,4593	14,1244	14,4480	12,7117	14,5040	13,0375	15,0218
$\gamma = 0.001$	Algoritmo 3	13,1599	13,9712	12,9908	13,4593	14,1244	14,4480	12,7117	14,5040	13,0375	15,0218

		Sesión									
		11	12	13	14	15	16	17	18	19	20
	Algoritmo 1	14,1310	12,7120	13,4788	15,2050	12,3617	13,3392	13,5788	14,8850	13,0147	13,4080
	Algoritmo 2	14,1129	12,6818	13,5222	15,3588	12,4230	13,3676	13,6076	14,8709	13,0250	13,4763
$\gamma = 0.1$	Algoritmo 3	14,0899	12,6811	13,4995	15,3693	12,3871	13,3549	13,6220	14,8512	13,0197	13,4378
$\gamma = 0.01$	Algoritmo 3	14,0897	12,6809	13,5000	15,3692	12,3864	13,3564	13,6190	14,8513	13,0199	13,4361
$\gamma = 0.001$	Algoritmo 3	14,0897	12,6809	13,5000	15,3692	12,3864	13,3564	13,6190	14,8513	13,0199	13,4361

Tabla 12. Precios por sesión caso 3

En la Figura 4, se puede observar de manera gráfica en que algoritmo se obtiene menores precios por sesión.

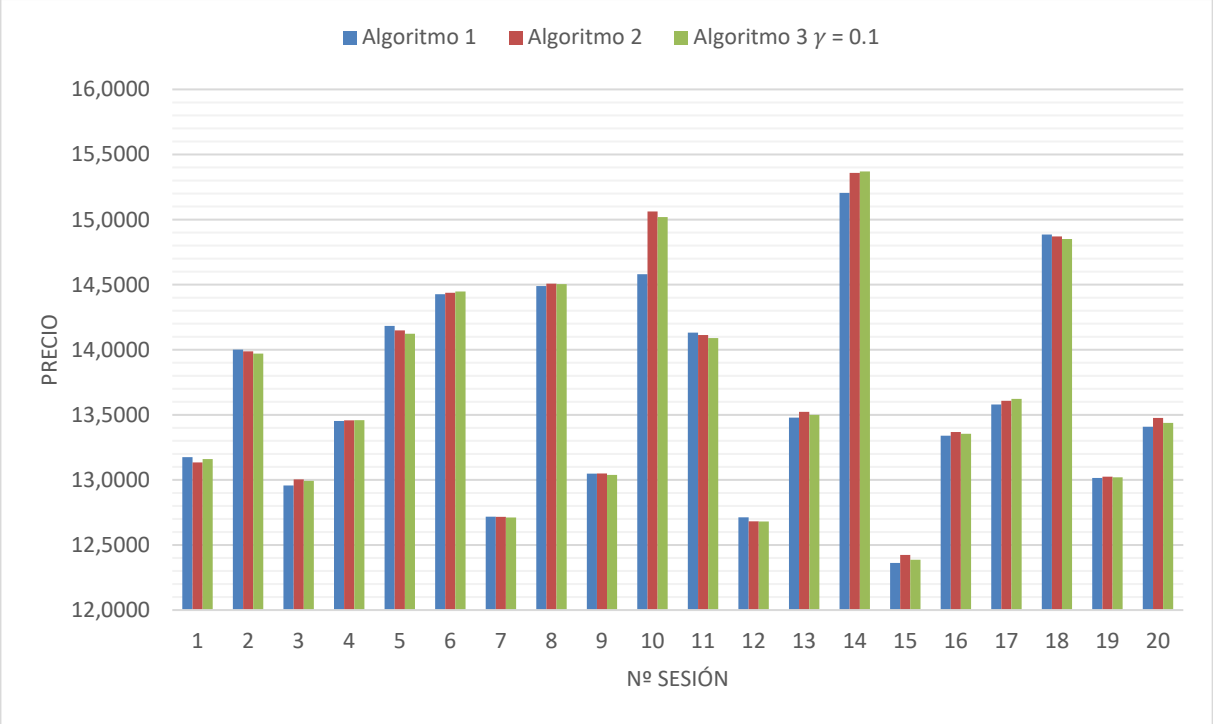

Figura 4. Gráfica de precios caso 3

Al igual que en el resto de casos, el precio de compra que proporciona el Algoritmo 1 es menor.

	Precio medio Algoritmo 1	13,6571
	Precio medio Algoritmo 2	13,6857
γ = 0.1	Precio medio Algoritmo 3	13,6869
γ = 0.01	Precio medio Algoritmo 3	13,6869
γ = 0.001	Precio medio Algoritmo 3	13,6869

Tabla 13. Precios medio caso 3

5.2 Resultados con horizonte de 40 minutos

Tras los resultados obtenidos en el apartado anterior, se ha decidido reducir el **horizonte** de 120 minutos a 40 minutos. Es decir, las sesiones de compra de acciones se realizarán en 40 minutos.

Es necesario, por tanto, modificar la Base de datos, DB, que tendrá las misma cantidad de sesiones, pero con menos minutos, y, por tanto, su secuencia óptima asociada también será de 40 minutos.

Entonces la base de datos tendrá ahora de dimensión: (1500x81)

- 1500 correspondientes al número de sesiones,
- y 81 se divide en, 40 correspondiente a los precios por minutos, (40 minutos como tamaño máximo de regresor), 1 componente asociada al volumen y luego por otro lado 40 subórdenes (decisiones de compra/venta), de acuerdo con esos 40 minutos.

Con este cambio, se reduce la aleatoriedad con la que se combinaban las secuencias óptimas para obtener la secuencia $\widehat{\boldsymbol{m}}^*$ que luego se multiplicará por los precios futuros, al tener que sumar 2.000.000 de acciones en 40 minutos y no 2.000.000 acciones en 120 minutos.

5.2.1 Comparativa

- CASO 4:
  - Parámetros ajustables: El parámetro γ aparece en la tabla de precios.

	Parámetros		
	tam_regresor	N entradas	n_test sesiones
Algoritmo 1	40	-	20
Algoritmo 2	40	100	20
Algoritmo 3	40	100	20

Tabla 14. Parámetros ajustables caso 4

  - Precios por sesión: Apenas varían los precios en los 2 últimos casos.

		Sesión									
		1	2	3	4	5	6	7	8	9	10
	Algoritmo 1	13,2063	14,0710	13,0047	13,4819	14,2286	14,4580	12,7000	14,4636	13,1242	15,1060
	Algoritmo 2	13,2016	13,9939	12,9763	13,4785	14,2186	14,4377	12,7226	14,4840	13,0813	15,1263
$\gamma = 0.1$	Algoritmo 3	13,2015	13,9996	12,9760	13,4787	14,2189	14,4374	12,7233	14,4860	13,0818	15,1273
$\gamma = 0.01$	Algoritmo 3	13,2014	14,0050	12,9751	13,4786	14,2190	14,4369	12,7238	14,4864	13,0813	15,1275
$\gamma = 0.001$	Algoritmo 3	13,2014	14,0050	12,9751	13,4786	14,2190	14,4369	12,7238	14,4864	13,0813	15,1275

		Sesión									
		11	12	13	14	15	16	17	18	19	20
	Algoritmo 1	14,0067	12,6270	13,5410	15,3670	12,3453	13,3440	13,6024	15,0037	12,9931	13,4090
	Algoritmo 2	14,0207	12,6326	13,5406	15,4230	12,3517	13,4389	13,5773	14,9089	12,9858	13,3622
$\gamma = 0.1$	Algoritmo 3	14,0196	12,6324	13,5410	15,4225	12,3528	13,4404	13,5736	14,9090	12,9854	13,3623
$\gamma = 0.01$	Algoritmo 3	14,0187	12,6323	13,5411	15,4226	12,3535	13,4414	13,5715	14,9090	12,9848	13,3620
$\gamma = 0.001$	Algoritmo 3	14,0187	12,6323	13,5411	15,4226	12,3535	13,4414	13,5715	14,9090	12,9848	13,3620

Tabla 15. Precios por sesión caso 4

En la Figura 5, se puede observar de manera gráfica en que algoritmo se obtienen menores precios por sesión.

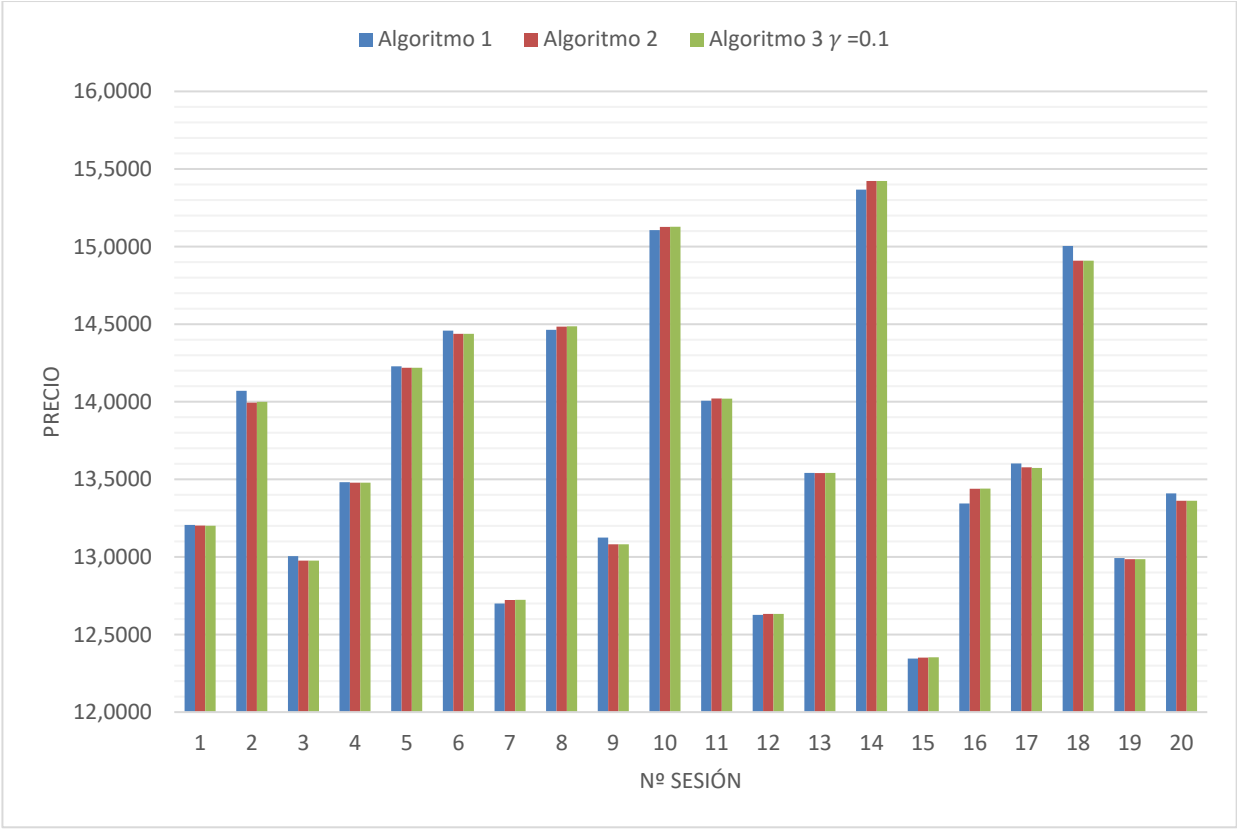

Figura 5. Gráfica de precios caso 4

- Precios medios: Se puede observar en la Tabla 16, que se obtienen mejores precios de compra en el Algoritmo 2. En el Algoritmo 3 se fuerza a que se combinen los estados más cercanos, y es más difícil encontrar una solución mejor.

	Precio medio Algoritmo 1	13,7042
	Precio medio Algoritmo 2	13,6981
γ = 0.1	Precio medio Algoritmo 3	13,6985
γ = 0.01	Precio medio Algoritmo 3	13,6986
γ = 0.001	Precio medio Algoritmo 3	13,6986

Tabla 16. Precios medios caso

- CASO 5:
  - Parámetros ajustables: Se ha aumentado únicamente el número de lambdas a 150.

	Parámetros		
	tam_regresor	N entradas	n_test sesiones
Algoritmo 1	40	-	20
Algoritmo 2	40	150	20
Algoritmo 3	40	150	20

Tabla 17. Parámetros ajustables caso 5

  - Precios por sesión: Igual que en el anterior caso apenas se producen cambios en los 2 últimos casos.

		Sesión									
		1	2	3	4	5	6	7	8	9	10
	Algoritmo 1	13,2063	14,0710	13,0047	13,4819	14,2286	14,4580	12,7000	14,4636	13,1242	15,1060
	Algoritmo 2	13,2016	14,0043	12,9778	13,4756	14,2155	14,4364	12,7222	14,4810	13,0665	15,1185
γ = 0.1	Algoritmo 3	13,2027	14,0046	12,9792	13,4769	14,2161	14,4377	12,7223	14,4836	13,0667	15,1185
γ = 0.01	Algoritmo 3	13,2035	14,0062	12,9798	13,4777	14,2163	14,4380	12,7221	14,4848	13,0703	15,1188
γ = 0.001	Algoritmo 3	13,2037	14,0065	12,9798	13,4779	14,2164	14,4379	12,7220	14,4847	13,0708	15,1189

		Sesión									
		11	12	13	14	15	16	17	18	19	20
	Algoritmo 1	14,0067	12,6270	13,5410	15,3670	12,3453	13,3440	13,6024	15,0037	12,9931	13,4090
	Algoritmo 2	14,0174	12,6366	13,5391	15,4347	12,3480	13,4211	13,5835	14,9336	12,9859	13,3526
γ = 0.1	Algoritmo 3	14,0166	12,6359	13,5410	15,4403	12,3467	13,4123	13,5823	14,9337	12,9856	13,3514
γ = 0.01	Algoritmo 3	14,0158	12,6357	13,5427	15,4395	12,3461	13,4109	13,5796	14,9337	12,9845	13,3515
γ = 0.001	Algoritmo 3	14,0155	12,6357	13,5430	15,4394	12,3460	13,4107	13,5787	14,9337	12,9842	13,3515

Tabla 18. Precios por sesión caso 5

En la Figura 6, se puede observar de manera gráfica en que algoritmo se obtiene menores precios por sesión.

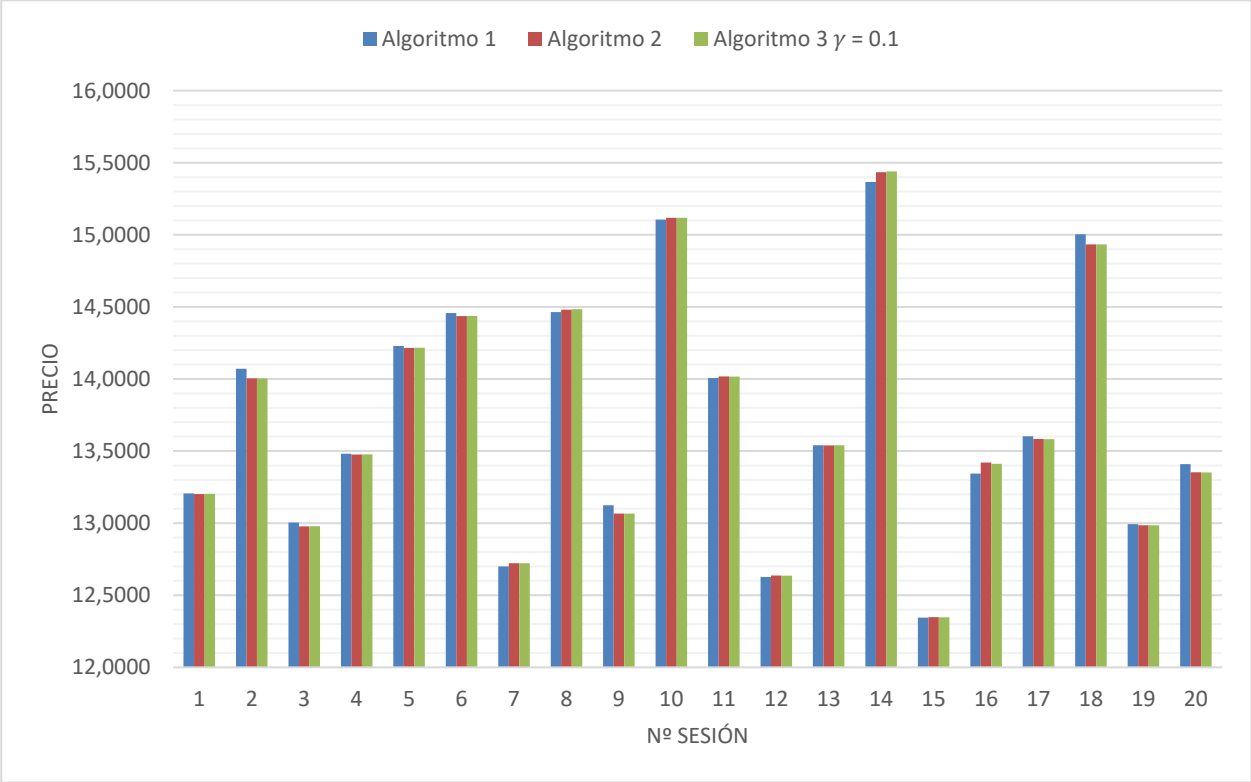

Figura 6. Precios por sesión caso 5

- Precios medios: Al igual que en el anterior caso, se ha obtenido un mejor precio para el Algoritmo 2.

	Precio medio Algoritmo 1	13,7042
	Precio medio Algoritmo 2	13,6976
γ = 0.1	Precio medio Algoritmo 3	13,6977
γ = 0.01	Precio medio Algoritmo 3	13,6979
γ = 0.001	Precio medio Algoritmo 3	13,6978

Tabla 19. Precios medio caso 5

- CASO 6:
  - Parámetros ajustables: Se aumenta de 150 entradas a 200 entradas.

	Parámetros		
	tam_regresor	N entradas	n_test sesiones
Algoritmo 1	40	-	20
Algoritmo 2	40	200	20
Algoritmo 3	40	200	20

Tabla 20. Parámetros ajustables caso 6

  - Precios por sesión: En este caso a diferencia de los 2 anteriores, si varían los precios para los 2 últimos valores de gamma.

		Sesión									
		1	2	3	4	5	6	7	8	9	10
	Algoritmo 1	13,2063	14,0710	13,0047	13,4819	14,2286	14,4580	12,7000	14,4636	13,1242	15,1060
	Algoritmo 2	13,2023	13,9982	12,9686	13,4768	14,2134	14,4448	12,7231	14,4841	13,0821	15,1220
$\gamma = 0.1$	Algoritmo 3	13,2021	14,0010	12,9717	13,4777	14,2157	14,4427	12,7227	14,4854	13,0825	15,1189
$\gamma = 0.01$	Algoritmo 3	13,2023	14,0010	12,9731	13,4784	14,2163	14,4408	12,7223	14,4859	13,0819	15,1183
$\gamma = 0.001$	Algoritmo 3	13,2023	14,0011	12,9733	13,4785	14,2163	14,4403	12,7222	14,4858	13,0819	15,1182

		Sesión									
		11	12	13	14	15	16	17	18	19	20
	Algoritmo 1	14,0067	12,6270	13,5410	15,3670	12,3453	13,3440	13,6024	15,0037	12,9931	13,4090
	Algoritmo 2	14,0075	12,6345	13,5191	15,4017	12,3485	13,4511	13,5858	14,9502	12,9867	13,3636
$\gamma = 0.1$	Algoritmo 3	14,0113	12,6349	13,5231	15,4120	12,3476	13,4415	13,5844	14,9491	12,9866	13,3588
$\gamma = 0.01$	Algoritmo 3	14,0120	12,6351	13,5250	15,4151	12,3472	13,4371	13,5818	14,9490	12,9858	13,3551
$\gamma = 0.001$	Algoritmo 3	14,0122	12,6352	13,5253	15,4152	12,3472	13,4365	13,5811	14,9490	12,9855	13,3547

Tabla 21. Precios por sesión caso 6

En la Figura 7, se puede observar de manera gráfica en que algoritmo se obtiene menores precios por sesión.

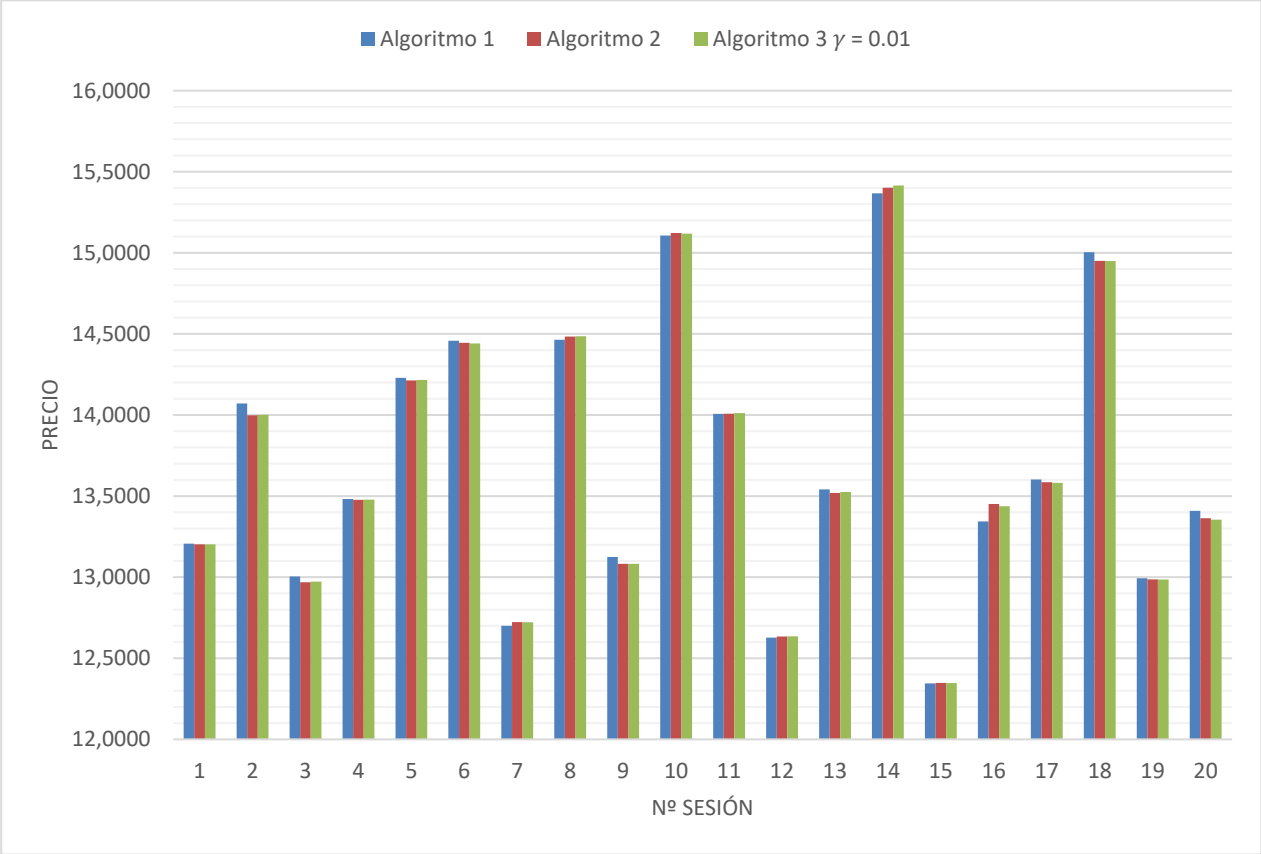

Figura 7. Precios por sesión caso 6

- Precio medio: En este caso se ha obtenido los mismos resultados en el Algoritmo 2 y en el Algoritmo 3 (γ = 0.01, γ = 0.001), esto es debido al incremento de N. Al haber tantos lambdas, aun teniendo que ponderar aquellos estados más cercanos, se ha conseguido el mismo precio.

	Precio medio Algoritmo 1	13,7042
	Precio medio Algoritmo 2	13,6982
γ = 0.1	Precio medio Algoritmo 3	13,6985
γ = 0.01	Precio medio Algoritmo 3	13,6982
γ = 0.001	Precio medio Algoritmo 3	13,6982

Tabla 22. Precios medios caso

6 CONCLUSIONES

En primer lugar, quisiera recordar cuáles eran los objetivos principales del proyecto:

- Obtener un algoritmo alternativo al que se llevó a cabo en [1].
- Verificar su viabilidad.
- Compararlo con el algoritmo de partida [1].

Tras obtener los resultados se han obtenido las siguientes conclusiones:

El Algoritmo 2 funcionaba perfectamente y aportaba valores muy parecidos a los del algoritmo [1]. Aunque era necesario realizar un ajuste porque la manera en la que combinaba los estados estaba lejos de la realidad. No tenía sentido que utilizara todos los estados de igual manera cuando estos distaban una gran longitud (distancia euclídea) del estado inicial.

Tras su modificación, y obtención del Algoritmo 3, el resultado de los precios no mejoró, algo lógico, ya que antes se obtenía el estado inicial combinando todos los estados para minimizar la función de coste. Tras añadir los términos de distancia euclídea en la función objetivo para ponderar más aquellos estados más cercanos al estado inicial, es imposible obtener unos resultados mejores que los anteriores debido a que se han acotado las posibles soluciones.

Otro aspecto importante ha sido el cambio de horizonte, reduciendo éste de 120 minutos a 40 minutos.

Cuando el horizonte estaba fijado en 120 minutos, los algoritmos debían comprar 2.000.000 de acciones en ese intervalo de tiempo. Cuando este combinaba las secuencias óptimas asociadas de los estados de la base de datos, la combinación tiene el mismo fundamento que la obtención del estado inicial, y eso no significa que sea la más eficiente, pues puede distribuir las 2.000.000 de acciones para los siguientes 120 minutos de una manera peor, ya que no conocemos los precios futuros, y, por lo tanto, puede dar un precio más alto de compra.

Tras reducir el horizonte a 40 minutos, se volvió a calcular los resultados de los algoritmos, obteniendo en este caso, un resultado favorable al Algoritmo 2 y el Algoritmo 3. Pues el algoritmo [1], distribuye la secuencia óptima de los estados de una manera muy regular, es decir, los valores de compra son muy parecidos en muchos minutos, mientras que la combinación de las secuencias optimas repartidas en los 40 minutos en el Algoritmo 2 y Algoritmo 3, han proporcionado un precio menor de compra.

Entre estos dos últimos, debe obtenerse un peor precio en el Algoritmo 3, ya que se fuerza a que combine con mayor ponderación aquellos estados más cercanos, mientras que en el Algoritmo 2, tiene más libertad, y es por esto que puede obtener menores precios, aunque esto no ocurra siempre así.

Durante la exposición de los resultados del horizonte de 120 minutos, se realizó una variación progresiva de los parámetros para concluir cuáles eran los ideales. Por ejemplo, a la hora de definir N, (el número de estados de la lista ordenada más parecidos al estado inicial), en la mayoría de los casos, los algoritmos proporcionaban mejores resultados cuándo se incrementaba N.

Pero en los resultados del horizonte de 40 minutos no ocurría así, se obtienen mejores valores para un N intermedio, ni muy grande, ni muy pequeño, el justo. En el proyecto, se ve reflejado para N=150, este valor obtiene mejores resultados que para N=100, o N=200. Y para el parámetro de ponderación γ, en todos los resultados coincide un mejor precio para γ=0.1, excepto para $N = 200$.

Para concluir, los objetivos han sido logrados de manera satisfactoria. Además, en cuánto a líneas futuras este proyecto es totalmente modificable, y se pueden añadir diferentes parámetros que ajusten aún más los resultados a la realidad.

REFERENCIAS

[1] G. ALFONSO, A. D. CARNERERO, D. R. RAMIREZ y T. ALAMO, «Receding Horizon Optimization of Large Trade Orders,» *IEEE Acces,* 2021.

[2] G. ALFONSO, A. D. CARNERERO, D. R. RAMIREZ y T. ALAMO, «Stock Forecasting Using Local Data,» *IEEE Acces,* 2020.

[3] H. White, «Economic preditction using neural networks: The case of IBM daily stock returns.,» *ICNN,* vol. Vol 2, pp. 451-458, 1988.

[4] D. Hasibuan, I. K. Jaya, B. Rumahorbo, J. Naibaho, J. Napitupulu y E. Rajagukguk, «Time series financial market forecasting based on support regression algorithm,» *IEEE (ICoSNIKOM),* pp. 1-4, 2010.

[5] J. Wang, T. Sun, B. Liu, Y. Cao y D. Wang, «Financial markets prediction with deep learning,» *IEEE (ICMLA),* pp. 97-104, 2018.

ANEXO: CÓDIGOS USADOS

Algoritmo 1: Receding Horizon Optimization of Large Trade Orders

```
clc;
clear; close all;

Nc= 120;

for flag_limit=0:0

    for flag_precio=0:1

        for n_dataset=5:5

        if n_dataset==0
            load('002040_reducido.mat');
        elseif n_dataset==1
            load('300239v5_reducido.mat')
        elseif n_dataset==2
            load('300033v2_reducido.mat')
        elseif n_dataset==3
            load('000001v2_mitad1_183000')
        elseif n_dataset==4
             load('002466v2_reducido.mat')
        elseif n_dataset==5
             load('000001v2_mitad2_183000_paraestado')
        elseif n_dataset==6
             load('000001v2_reducido.mat')
        elseif n_dataset==7
             load('000333v2_reducido.mat')
        elseif n_dataset==8
             load('600519v2_reducido.mat')
        end

        n_test = 1500;


        num_acciones_total = 20000; %Las acciones van en paquetes

        %Es decir, se compran 20000 paquetes de 100.

                    alpha = 0.1;
                    beta = 1.1;

                    mu = 0.65;
                    sigma = 1.05;

        tic

        %Estoy suponiendo que num_acciones son los paquetes de acciones
        num_acciones = num_acciones_total;
```

```
tam_regresor = 100;

X_Price = [X_Price(:,1:tam_regresor) X_Volume(:,1)];

X_Volume = [X_Volume(:,1:tam_regresor) X_Price(:,1)];

for n=1:n_test

    u_hist = [];

    num_acciones = num_acciones_total;
    Nc = 120;

    X_Price_test = X_Price_test_h(:,:,n);
    Y_Price_test = Y_Price_test_h(:,:,n);

    X_Price_real_test = X_Price_real_test_h(:,:,n);
    Y_Price_real_test = Y_Price_real_test_h(:,:,n);

    X_Volume_test = X_Volume_test_h(:,:,n);
    Y_Volume_test = Y_Volume_test_h(:,:,n);

    X_Volume_real_test = X_Volume_real_test_h(:,:,n);
    Y_Volume_real_test = Y_Volume_real_test_h(:,:,n);


    X_Price_test = [X_Price_test(:,1:tam_regresor) X_Volume_test(:,1)];
    X_Volume_test = [X_Volume_test(:,1:tam_regresor) X_Price_test(:,1)];

    TWAP = mean(Y_Price_real_test(1,:));

    %Tenemos que decidir cómo se va a elegir el límite

  %  limit = 100*(0.995*TWAP);
    limit = 100000000;

    for k=1:120

    %Aquí vendría la predicción del precio y de los volúmenes
      if flag_precio==0

          precio = DirectWeight(X_Price,Y_Price_real,X_Price_test(k,:),350);
          precio = precio(1:Nc)';

      else
          precio = Y_Price_real_test(1,k:end)';
      end

        precio = precio*100;

     if flag_precio==0

      volumen = DirectWeight(X_Volume,Y_Volume_real,X_Volume_test(k,:),400);
      volumen = volumen(1:Nc)';

      volumen(volumen<0) = abs(volumen(1));
```

```
else
  volumen = DirectWeight(X_Volume,Y_Volume_real,X_Volume_test(k,:),400);
  volumen = volumen(1:Nc)';

  volumen(volumen<0) = abs(volumen(1));

  %volumen = Y_Volume_real_test(1,k:end)';
end

    volumen = volumen/100;

    %Hago la modificación pertinente

    marca = precio<limit;
    %
    marca(1) = 1;
    marca_num = find(marca==1);

    Nc_reducido = sum(marca==1);

    %Voy a comprobar que puedo comprar todas mis acciones para el
    %horizonte reducido

    if Nc_reducido*(0.1*num_acciones_total)<num_acciones

        m = 0.1*num_acciones_total*ones(1,Nc_reducido);
        u(marca_num) = m;

        flag_solver = 0;
    else
        num_acciones_reducido = num_acciones;

        flag_solver = 1;
    end

    %Digamos que estamos comprando
    a = 1;

    if flag_solver==1

        cota = precio<=limit;
        cota(1) = 1;
        cota = 0.1*num_acciones_total*cota;

        d = a*precio + mu*sigma.^((1:Nc)');
        e = (alpha)./(volumen.^beta);

        lambda_min = min(d);

        aux_max = (beta+1)*(cota.^beta).*e + d;

        lambda_max = max(aux_max);

        lambda = (lambda_max+lambda_min)/2;

        lambda_a = lambda_min;
        lambda_b = lambda_max;

        flag = 0;
```

```
            while flag==0

                 m_prueba =  (-precio-(mu*sigma.^((1:Nc)'))+lambda).
                /((beta+1).*(alpha./(volumen.^beta)));
                m = ((-d+lambda)./((beta+1).*e)).^(1/beta);

                m(m<0) = 0;

                aux_cota = m>cota;

                m(aux_cota) = cota(aux_cota);

                if(sum(m)<num_acciones_reducido+1e-2 &&
                sum(m)>num_acciones_reducido-1e-2)

                    flag=1;
                end

                if(sum(m)>num_acciones_reducido)
                    %Me he pasado
                    lambda_b = lambda;
                    lambda = (lambda+lambda_a)/2;
                else
                    %Me he quedado corto
                    lambda_a = lambda;
                    lambda = (lambda_b+lambda)/2;
                end

            end

            u = round(m);

        end

        if Y_Price_real_test(1,k)<=(limit/100)
            num_acciones = num_acciones-u(1);
            u_hist(k,:) = u(1);
        else
            u_hist(k,:) = 0;
        end

        Nc = Nc-1;
        precio = precio(2:end);

        u0 = u(1);

        volumen = volumen(2:end);

        if num_acciones==0 || Nc==0

            u_hist = [u_hist; zeros(Nc,1)];
            break;

        end
    end
```

```
            aux = u_hist.*Y_Price_real_test(1,:)';
            Precio_medio_compra_real = sum(aux(aux>0))/sum(u_hist);

            Precio_mercado_real = mean(Y_Price_real_test(1,:));

            Precio_VWAP =
            sum(Y_Volume_real_test(1,:).*Y_Price_real_test(1,:))/sum(Y_Volume_real_t
            est(1,:));

            u_hist_h(n,:) = u_hist';

            Precio_compra_h(n) = Precio_medio_compra_real;
            Precio_VWAP_h(n) = Precio_VWAP;
            Precio_TWAP_h(n) = Precio_mercado_real;

        end

            Resultados_VWAP(n_dataset+1) = mean(Precio_VWAP_h);
            Resultados_TWAP(n_dataset+1) = mean(Precio_TWAP_h);

        if flag_precio==1
                Resultados_compra_real(n_dataset+1) = mean(Precio_compra_h);
        else
                Resultados_compra_prediccion(n_dataset+1) = mean(Precio_compra_h);
        end

                 clearvars -except n_dataset flag_limit flag_precio...
                 Resultados_VWAP Resultados_TWAP Resultados_compra_real...
                 Resultados_compra_prediccion

        end

    end

end

          disp('%Esto es la tabla de todos los Precios ')
          disp('')

                   disp('\begin{center}')
        disp('\begin{tabular}{ c c c c c c c c}')
        for kk=1:9
            if kk<9
          cadena = [num2str(kk-1),' & ',num2str(Resultados_compra_real(kk)),' &
',...
              num2str(Resultados_compra_prediccion(kk)),' & ',...
              num2str(Resultados_VWAP(kk)),' & ',...
              num2str(Resultados_TWAP(kk)),' \\ '];
            else
          cadena = [num2str(kk-1),' & ',num2str(Resultados_compra_real(kk)),' &
',...
              num2str(Resultados_compra_prediccion(kk)),' & ',...
              num2str(Resultados_VWAP(kk)),' & ',...
              num2str(Resultados_TWAP(kk))];
            end
          disp(cadena)
        end
        disp('\end{tabular}')
        disp('\end{center}')
        disp('');
```

Algoritmo 2: Optimización mediante estimación/aprendizaje de la partición de grandes órdenes

```
%Algoritmo estimacion de secuencia optima
load('000001v2_mitad2_183000_paraestado')
load('BD1500_000001'); %cargo base de datos
% Inicializo-------------------------------------

BDx = zeros(1500,1); %Variable auxiliar, me sirve para ordenar
posicion = zeros(1500,1);
BDordenada = zeros(1500,221);  %sera la base de datos ordenada
tam_regresor = 100;
n_test= 20;  % (n sesiones)
k_horizonte= 120;
Precio = zeros(1,n_test);
N = 200; % N primeras entradas de la lista BD mas parecidas al estado del mercado
(numero de lambdas)
lambda = zeros(N,1);
tam_2 = tam_regresor;
x0 = zeros(N,1);
gamma = 0.001;




for n=1:n_test
    for k=1:1

        %X_Price_test = X_Price_test_h(:,:,n); %vector de estado
        %X_Volume_test = X_Volume_test_h(:,:,n);

        X_Price_test =  X_Price_test_h(1,1:tam_regresor,n) ;

%Paso 1 algoritmo-------------------------------------------------------

        for m=1:1500
            %distancia euclidea entre Zi(t) y BD(z)
            BDx(m) = norm(X_Price_test(k,1:tam_2)-BD(m,1:tam_2)) ;
            posicion(m) = m;
        end


%Paso 2 algoritmo-------------------------------------------------------

        [BD2,I] = sort(BDx); %Ordeno segun distancia
        Ysorted = posicion(I);
        for j=1:1500
            BD2(j,2) = Ysorted(j);
        end

        for i=1:1500
            BDordenada(i,:) = BD(Ysorted(i),:); %Reescribo base de datos de manera
                                                  ordenada segun distancia
        end
```

```
%Paso 4 Programacion cuadrática-------------------------------------------

   %matrix Base de datos ordenada
    Aeq = [BDordenada(1:N,1:tam_regresor)'; ones(1,N)];
    beq = [X_Price_test(k,1:tam_regresor)' ; 1]; %vector estado del mercado

   %restriccion para m sea no negativa, y cada paquete <20000
    A = [-BDordenada(1:N,102:221)'; BDordenada(1:N,102:221)'] ;

    b = [zeros(k_horizonte,1) ; 2000  *ones(k_horizonte,1)]; %vector

   H = 2*eye(N);
    f = zeros(N,1); %debe ser 0
    lb = zeros(N,1);
    ub = ones(N,1);

      opts = optimoptions(@quadprog,'MaxIterations',3500,'OptimalityTolerance',1e-
      7,'ConstraintTolerance',1e-7);
    [lambda(:,:),fval,exitflag,output] = quadprog(H,f,A,b,Aeq,beq,[],[],[],opts);

%Paso 5 calcular m(t)-----------------------------

            m = transpose(lambda'*BDordenada(1:N,102:221)); %vector de compra

%Simulacion-----------------------------------------

    if exitflag == 1 %Cuando la optimizacion encuentra solucion optima
        Precio(n) = Y_Price_real_test_h(1,1:k_horizonte,n)*m/20000; %suma del precio
total de las acciones compradas en una sesion
    else
        Precio(n) = 0;
    end
    end
end
```

Algoritmo 3: Optimización mediante estimación/aprendizaje de la partición de grandes órdenes con ponderación de la Distancia Euclidea

```
%Algoritmo estimacion de secuencia optima
load('000001v2_mitad2_183000_paraestado')
load('BD_Horizonte_Reducido_40'); %cargo base de datos
% Inicializo-----------------------------------

BDx = zeros(1500,1); %Variable auxiliar, me sirve para ordenar
posicion = zeros(1500,1);
BDordenada = zeros(1500,81);  %sera la base de datos ordenada
tam_regresor = 40;
n_test= 20;  % (n sesiones)
k_horizonte= 40;
Precio = zeros(1,n_test);
N = 200; % N primeras entradas de la lista BD mas parecidas al estado del mercado
(numero de lambdas)
lambda = zeros(N,1);
tam_2 = tam_regresor;
x0 = zeros(N,1);
gamma = 0.001;
```

```
for n=1:n_test
    for k=1:1

        %X_Price_test = X_Price_test_h(:,:,n); %vector de estado
        %X_Volume_test = X_Volume_test_h(:,:,n);

        X_Price_test =  X_Price_test_h(1,1:tam_regresor,n) ;

%Paso 1 algoritmo-----------------------------------------------------

        for m=1:1500
            %distancia euclidea entre Zi(t) y BD(z)
            BDx(m) = norm(X_Price_test(k,1:tam_2)-BD(m,1:tam_2)) ;
            posicion(m) = m;
        end


%Paso 2 algoritmo-----------------------------------------------------

        [BD2,I] = sort(BDx); %Ordeno segun distancia
        Ysorted = posicion(I);
        for j=1:1500
            BD2(j,2) = Ysorted(j);
        end

        for i=1:1500
            BDordenada(i,:) = BD(Ysorted(i),:); %Reescribo base de datos de manera
                                                  ordenada segun distancia
        end




%Paso 4 Programacion cuadrática-----------------------------------------

   %matrix Base de datos ordenada
    Aeq = [BDordenada(1:N,1:tam_regresor)'; ones(1,N)];
    beq = [X_Price_test(k,1:tam_regresor)' ; 1]; %vector estado del mercado

   %restriccion para m sea no negativa, y cada paquete <20000
    A = [-BDordenada(1:N,42:81)'; BDordenada(1:N,42:81)'] ;

    b = [zeros(k_horizonte,1) ; 2000  *ones(k_horizonte,1)]; %vector

    H = 2*(diag(BD2(1:N,1)) + gamma*eye(N)); %matrix que arregla la funcion objetivo
    f = zeros(N,1); %debe ser 0
    lb = zeros(N,1);
    ub = ones(N,1);

      opts = optimoptions(@quadprog,'MaxIterations',3500,'OptimalityTolerance',1e-
      7,'ConstraintTolerance',1e-7);
    [lambda(:,:),fval,exitflag,output] = quadprog(H,f,A,b,Aeq,beq,[],[],[],opts);
```

```
%Paso 5 calcular m(t)-------------------------------

            m = transpose(lambda'*BDordenada(1:N,42:81)); %vector de compra




%Simulacion-----------------------------------------

    if exitflag == 1 %Cuando la optimizacion encuentra solucion optima
        Precio(n) = Y_Price_real_test_h(1,1:k_horizonte,n)*m/20000; %suma del precio
total de las acciones compradas en una sesion
    else
        Precio(n) = 0;
    end
    end
end
```

Cargar_muchos_datos_test_generico

```
clear;

file = input('Nombre del fichero excel: ','s');

file_complete = [file '.xlsx'];

%file='688012v2.xlsx';

[~,txt,raw]=xlsread(file_complete);
%Price = cell2mat(raw(2:end,2));
%Volume = cell2mat(raw(2:end,3));
Price = cell2mat(raw(2:end,3));
Volume = cell2mat(raw(2:end,4));
%day = cell2mat(raw(2:end,4));

% Price_test = Price(14632:end);
% Volume_test = Volume(14632:end);
% day_test = day(14632:end);
%

Price = Price(not(isnan(Price)));
Volume = Volume(not(isnan(Price)));
% day = day(1:14631);

% X_Price = [];
% Y_Price = [];
%
% X_Volume = [];
% Y_Volume = [];
%
% X_day = [];
% Y_day = [];
%
% Y_Price_real = [];
% Y_Volume_real = [];
%
```

```
% X_Price_real = [];
% X_Volume_real = [];

p=120;

Price_aux = Price;
Volume_aux = Volume;

Price = ExpMovingAverage(Price,5);
Volume = ExpMovingAverage(Volume,5);

min_Price = min(Price);
max_Price = max(Price);

min_Volume = min(Volume);
max_Volume = max(Volume);

Price = (Price-min(Price))/(max(Price)-min(Price));
Volume = (Volume-min(Volume))/(max(Volume)-min(Volume));

% RDP5=[];
% RDP10=[];

    Y_Price_real = zeros(length(Price)-(p-1)-241,120);
    Y_Volume_real = zeros(length(Price)-(p-1)-241,120);

    X_Price_real = zeros(length(Price)-(p-1)-241,120);
    X_Volume_real = zeros(length(Price)-(p-1)-241,120);

    Y_Price = zeros(length(Price)-(p-1)-241,120);
    X_Price = zeros(length(Price)-(p-1)-241,120);

    Y_Volume = zeros(length(Price)-(p-1)-241,120);
    X_Volume = zeros(length(Price)-(p-1)-241,120);

%     Y_day = zeros(length(Price)-(p-1)-241,120);
%     X_day = zeros(length(Price)-(p-1)-241,120);




for ind = 241:length(Price)-(p-1)
    Y_Price_real(ind-240,:)  = Price_aux(ind:ind+(p-1))';
    Y_Volume_real(ind-240,:) = Volume_aux(ind:ind+(p-1))';

    X_Price_real(ind-240,:) = Price_aux(ind-1:-1:ind-120)';
    X_Volume_real(ind-240,:) = Volume_aux(ind-1:-1:ind-120)';

    Y_Price(ind-240,:) = Price(ind:ind+(p-1))';
    X_Price(ind-240,:) = Price(ind-1:-1:ind-120)';

    Y_Volume(ind-240,:) = Volume(ind:ind+(p-1))';
    X_Volume(ind-240,:) = Volume(ind-1:-1:ind-120)';
```

```
%     Y_day(ind-240,:) = day(ind:ind+(p-1))';
%     X_day(ind-240,:) = day(ind-1:-1:ind-120)';

    %RDP5 = [RDP5; (Datos_ax(ind-1)-Datos_ax(ind-6))/Datos_ax(ind-6)];
    %RDP10 = [RDP10; (Datos_ax(ind-1)-Datos_ax(ind-11))/Datos_ax(ind-11)];
end



n_test = 100;

aux = randperm(floor(length(X_Price)/120));
aux = aux(1:n_test);
index =[];

for i=1:n_test

    index = [index; (120*(aux(i)-1)+1:120*(aux(i)))'];

    X_Price_test_h(:,:,i) = X_Price(120*(aux(i)-1)+1:120*(aux(i)),:);
    Y_Price_test_h(:,:,i) = Y_Price(120*(aux(i)-1)+1:120*(aux(i)),:);

    X_Price_real_test_h(:,:,i) = X_Price_real(120*(aux(i)-
1)+1:120*(aux(i)),:);
    Y_Price_real_test_h(:,:,i) = Y_Price_real(120*(aux(i)-
1)+1:120*(aux(i)),:);

    X_Volume_test_h(:,:,i) = X_Volume(120*(aux(i)-1)+1:120*(aux(i)),:);
    Y_Volume_test_h(:,:,i) = Y_Volume(120*(aux(i)-1)+1:120*(aux(i)),:);

    X_Volume_real_test_h(:,:,i) = X_Volume_real(120*(aux(i)-
1)+1:120*(aux(i)),:);
    Y_Volume_real_test_h(:,:,i) = Y_Volume_real(120*(aux(i)-
1)+1:120*(aux(i)),:);

end

    X_Price(index,:) = [];
    Y_Price(index,:) = [];

    X_Price_real(index,:) = [];
    Y_Price_real(index,:) = [];

    X_Volume(index,:) = [];
    Y_Volume(index,:) = [];

    X_Volume_real(index,:) = [];
    Y_Volume_real(index,:) = [];

clearvars raw txt file_complete aux i ind index p Price Price_aux Volume
Volume_aux;

save(file);
```

DirectWeight

```
function Y_TEST = DirectWeight(X_DAT,Y_DAT,X_TEST,num_vec)
% Y_TEST = DirectWeight(X_DAT,Y_DAT,X_TEST)
%
%Función que hace el Direct Weight optimization para predicción.
%
%- X_DAT son las entradas pasadas.
%- Y_DAT son las salidas pasadas asociadas a X_DAT.
%Juntos forman el dataset equivalente al conjunto de training
%y validation en las NN.
%- X_TEST son las entradas que queremos predecir.
%- num_vec número de puntos vecinos en los que se va a apoyar la
%predicción. Mi recomendación es poner entre 250~500.
%
%Las dimensiones de las matrices tienen que ser las siguientes:
%- X_DAT = n_dataset*m     ->     Siendo n_dataset el número de puntos que
componen el
%dataset y m el tamaño del regresor.
%- Y_DAT = n_dataset*1     ->     Siendo n_dataset el número de puntos que
componen el
%dataset.
%- X_TEST = n_test*m       ->     Siendo n_test el número de puntos que
componen el
%conjunto de test y m el tamaño del regresor.
%- num_vec es un escalar.

if nargin~=4
    disp('Error: Hay que introducir 4 argumentos');
    return;
end

ind_Training = size(X_DAT,1);
ind_Test = size(X_TEST,1);

m = size(Y_DAT,2);

if ind_Training<num_vec
    disp('Error: El dataset debe tener al menos num_vec puntos');
    return;
end

num_pun = num_vec;

Y_TEST = zeros(ind_Test,m);


    for i=1:ind_Test

            x = X_TEST(i,:);

            X = X_DAT;
            Y = Y_DAT;
```

```
                [~,sel] = sort(sum((X_DAT-repmat(x,[ind_Training 1])).^2,2));

                X = X(sel(1:num_pun),:);
                Y = Y(sel(1:num_pun),:);

                Aeq = [X'; ones(1,num_pun)];
                beq = [x'; 1];

                %[res,~] = quadprog(2*eye(num_pun),[],[],[],Aeq,beq);

                   Q=2*eye(num_pun);
                E=Aeq;
                d=beq;
                [m,n]=size(d);

                mat=[Q E';E zeros(m,m)];
                term=[zeros(num_pun,1);d];
                res=mat\term;

                res=res(1:num_pun);


                Y_TEST(i,:) = Y'*res;

            end

        end
```