Trabajo Fin de Grado
Grado en Ingeniería de Organización Industrial

Optimización de la dinámica de una cadena de suministro circular mediante inventory controllers

Autor: Jesús Santamaría Vázquez

Tutor: Roberto Domínguez Cañizares

Dpto. Organización Industrial y Gestión de Empresas I
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2023

Trabajo Fin de Grado

Grado en Ingeniería de Organización Industrial

Optimización de la dinámica de una cadena de suministro circular mediante inventory controllers

Autor:

Jesús Santamaría Vázquez

Tutor:

Roberto Domínguez Cañizares

Profesor Titular

Dpto. Organización Industrial y Gestión de Empresas I

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2023

Trabajo Fin de Grado: Optimización de la dinámica de una cadena de suministro circular mediante inventory controllers

Autor: Jesús Santamaría Vázquez
Tutor: Roberto Domínguez Cañizares

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

Sevilla, 2023

El Secretario del Tribunal

Agradecimientos

Quiero expresar mi más sincero agradecimiento a mis profesores, familia y amigos por su apoyo y guía durante la elaboración de mi Trabajo Final de Grado. Sin su paciencia, sabiduría y motivación, no habría sido posible alcanzar este logro.

A mi familia y amigos, quiero agradecerles por su incondicional amor y apoyo. Gracias por ser mi roca y por estar siempre a mi lado, incluso en los momentos más difíciles.

Mis profesores, por su parte, me proporcionaron una sólida formación académica y me inspiraron a seguir aprendiendo y creciendo. Gracias por compartir su conocimiento y experiencia conmigo.

A mi tutor, Roberto, quiero expresarle mi más sincero agradecimiento por todo su apoyo y dedicación. Gracias por su guía, paciencia y consejos a lo largo de estos meses.

En definitiva, quiero agradecer a todos ustedes por su papel en mi vida y en mi formación como profesional. Este Trabajo Final de Grado es también un logro suyo, y estoy muy orgulloso de haber contado con su ayuda y apoyo.

Jesús Santamaría Vázquez

Sevilla, 2023

Índice

Índice de Tablas

Índice de Códigos

Índice de Figuras

Notación

1. Variables:

mc_t	Número de unidades nuevas que finalizan en el periodo "t"
rc_t	Número de unidades refabricadas que finalizan en el periodo "t"
is_t	Stock inicial en el periodo "t"
d_t	Demanda del cliente en el periodo “t”
r_t	Devoluciones del cliente en el periodo “t”
ns_t	Stock neto en el periodo “t”
w_t	Trabajo en curso en el periodo “t”
o_t	Orden de producción en el periodo “t”
$\widehat{d}_t$	Previsión de la demanda del periodo “t”
ss_t	Stock de seguridad del periodo “t”
tw_t	Trabajo en curso objetivo del periodo “t”

2. Parámetros de control:

α	Factor de suavización, determina la sensibilidad que debe tener la regla de previsión de demanda del modelo a las variaciones reales de esta.
ε	Nivel de protección contra la falta de existencias.
T_i	Constante de tiempo del controlador de inventario disponible
T_w	Constante de tiempo del controlador de inventario en proceso
T_p	Estimación del plazo de entrega

3. Parámetros incontrolables:

μ	Media de la distribución normal que define la demanda
CV_d	Coeficiente de variación de la demanda.

β	Media de la distribución normal que define la tasa de retorno.
CV_r	Coeficiente de variación de la tasa de retorno.
T_m	Tiempo medio de fabricación.
T_r	Tiempo medio de refabricación.
T_c	Tiempo medio de consumo.
k_o	Peso de la variabilidad de pedidos.
k_i	Peso de la variabilidad de inventario.

4. Indicadores de desempeño:

OVR	Ratio de variabilidad de pedidos.
IVR	Ratio de variabilidad de inventario.
J	Métrica de variabilidad total.

1 Introducción

Previo a la descripción del objetivo de este trabajo así como de la estructura que se va a seguir, se definen ciertos aspectos relacionados con las cadenas de suministro que facilitarán la comprensión del mismo.

1.1 Cadena de suministros

Se denomina cadena de suministro al conjunto de procesos y actividades que se llevan a cabo a lo largo del desarrollo de un producto o servicio, desde la obtención de materias primas hasta la entrega del producto o servicio en su estado final al consumidor, además del servicio de postventa y garantía en los casos que corresponda. El objetivo final de la cadena de suministro es satisfacer las necesidades de los clientes de la manera más eficiente posible.

Para alcanzar este objetivo, las cadenas de suministro deben cumplir una serie de principios o características:

1. **Recursos adecuados:** Se deben buscar proveedores que garanticen la disponibilidad de materias primas y recursos suficientes para la producción y entrega del producto, o la adecuada prestación de servicio al cliente final.

2. **En buenas condiciones:** Es necesario garantizar unas óptimas condiciones de producción, almacenamiento y transporte. El valor del producto o servicio percibido por el cliente define la imagen de la empresa.

3. **En las cantidades suficientes:** Se deben almacenar las existencias de modo que se pueda suplir siempre la demanda, pero de manera que se utilicen los mínimos recursos posibles, de inventario, mano de obra, etc.

4. **Dentro de los tiempos acordados:** Se debe buscar siempre minimizar los tiempos de espera, es decir, adoptar una filosofía "*lead time eficiente*", ya que en el mundo en el que se desarrolla el mercado actual, el cliente tiene la posibilidad de elegir el producto de la competencia en cualquier momento.

5. **Orientado al cliente:** Por mucho que se cumplan el resto de principios, es imprescindible que el producto o servicio supla las necesidades particulares del cliente, sin errores ni variaciones de ningún tipo.

En general, a lo largo de la cadena de suministro de un producto encontramos diferentes operaciones de transformación, fabricación y transporte intercaladas entre sí y con una secuencia establecida. Por ello, es importante planificar una estrategia para distribuir de forma adecuada la logística que todos estos procesos conllevan. Así como conseguir un flujo de productos, materiales e información fluido entre las diferentes operaciones que componen los eslabones de la cadena.

Sin embargo, debido a la propia complejidad de las cadenas de suministro, en ocasiones se dan fenómenos que merman su eficiencia y/o eficacia, como el efecto *Látigo* o *Bullwhip.*

1.2 Efecto *Látigo* o *Bullwhip.*

En una cadena convencional, cada uno de los eslabones se sostiene gracias a sus eslabones adyacentes y lo mismo ocurre con estos. En una cadena de suministro ocurre algo similar, cada uno de los eslabones hace pedidos periódicamente al eslabón anterior en función de la demanda del eslabón siguiente, o del cliente final en caso de tratarse del último eslabón de la cadena.

Los volúmenes de estos pedidos resultan de una estimación basada en diversos factores, como podrían ser los tiempos de aprovisionamiento o el histórico de demanda. Una vez calculados, las empresas suelen añadir algunas unidades extra al pedido como margen de seguridad, de manera que puedan cumplir con la demanda del siguiente eslabón en caso de que se dé un aumento puntual de esta. Al extender esta práctica a lo largo de toda la cadena, es fácil observar que cada eslabón incrementa el tamaño del pedido que ya fue aumentado por el siguiente eslabón de la cadena de suministro. Esto ocurre debido a que en cada eslabón se realizan pedidos en función de la demanda que ellos perciben, y en muchos casos esta demanda no es real, ya que se ve incrementada a causa de los márgenes o stocks de seguridad de los eslabones siguientes. Esta situación no suele presentar grandes problemas mientras la demanda del cliente final se mantenga a un nivel más o menos constante.

En cambio, en caso de darse un aumento puntual de la demanda del cliente final estaríamos ante el pulso inicial del efecto látigo. Este aumento de la demanda se iría incrementando cada vez más a lo largo de la cadena resultando en un volumen de fabricación sobredimensionado (en el punto final del látigo, asociado al inicio de la cadena de suministro), dando lugar inicialmente a un periodo de alta demanda para las fábricas, exigiéndoles capacidades de producción muy altas, y posteriormente periodos de baja demanda donde las fábricas quedan prácticamente inutilizadas. Podemos deducir por lo tanto, que este fenómeno causa entre otros los siguientes efectos: Mayores inversiones en inventarios totalmente innecesarias; situaciones de falta de stock, ya que los fabricantes dejan de producir para intentar paliar el sobredimensionamiento de los inventarios; fluctuaciones en los precios según la ley de oferta y demanda.

Para amortiguar los efectos de este fenómeno es importante que exista una comunicación fluida a lo largo de la cadena de suministro, que incluya una retroalimentación entre los diferentes eslabones de la cadena, así como un sistema de información transversal que nos permita observar el proceso mediante una visión global.

1.3 Cadena de suministro de ciclo cerrado (CSCC)

Las cadenas de suministro son responsables de la mayor parte de emisión de carbono al medio ambiente, contribuyendo así al calentamiento global y al cambio climático. Por otra parte, el ciclo de vida de los productos está disminuyendo a causa de la incesante evolución de la tecnología, lo que se traduce en un aumento de los residuos. Esto ha generado una mayor preocupación ambiental y ha llevado a un cambio en el enfoque de la gestión de las cadenas de suministro hacia la producción y preservación de los recursos de la tierra, motivado por la presión legislativa y los posibles beneficios económicos.

En este sentido se recurre a la logística inversa enfocándose en la recuperación de productos, a través de colectores y recicladores, y su posterior envío a los proveedores o fabricantes. Esta práctica es valiosa desde una perspectiva económica en cualquier sector industrial, además de ayudar a los participantes de la cadena de suministro a reducir su impacto ambiental. Al reducir el consumo de energía y las emisiones de gases de efecto invernadero, la logística inversa contribuye a la sostenibilidad global y a una mejor gestión de los recursos naturales.

Las cadenas de suministro que emplean este sistema de logística inversa reciben el nombre de Cadenas de Suministro de Ciclo Cerrado o Cadenas de suministro circulares, Bouchery et al. (2017) lo definen como:

"Una serie de procesos y flujos destinados a alguna forma de reutilización o recuperación de productos y materiales. En concreto, una cadena de suministro de circuito cerrado incorpora el diseño, el control y el funcionamiento de un sistema para maximizar la creación de valor a lo largo de todo el ciclo de vida de un producto, con una recuperación dinámica del valor a partir de diferentes tipos y volúmenes de devoluciones a lo largo del tiempo." Bouchery et al. (2017)

La configuración de este tipo de cadenas de suministro puede ser muy variada, dependiendo fundamentalmente de las necesidades de las diferentes empresas, principalmente se pueden clasificar en función del tipo o del nivel de reciclaje.

Según el tipo de reciclaje:

- Reciclaje interno: se refiere al proceso de recuperación de materiales o componentes de un producto en el mismo proceso de producción, de manera que los materiales no se pierden y pueden ser reutilizados en la fabricación de nuevos productos.

- Reciclaje externo: se trata del proceso de recuperación de materiales de productos al final de su vida útil, de manera que los materiales puedan ser utilizados para producir nuevos productos o para generar energía.

- Reutilización: en este caso, los productos se diseñan para que puedan ser utilizados de nuevo en su forma original o con algunas modificaciones mínimas.

- Remanufactura: implica la recuperación de componentes de un producto para ser utilizados en la fabricación de un producto nuevo.

Según el nivel de reciclaje:

- En primer lugar, podemos encontrar cadenas de suministro destinadas única y exclusivamente al reciclado de productos o materiales, surgen de una configuración centralizada del sistema de logística inversa, en la que ciertas entidades se encargan de todos los flujos de retorno de una empresa o de un grupo empresarial, siendo subcontratadas por estos como proveedores externos de logística inversa.

- Por otro lado, en otras ocasiones no se dan los condicionantes adecuados para establecer una configuración centralizada del sistema de logística inversa, y se opta por una configuración descentralizada. En estos casos, las cadenas de suministro deben optar por un modelo combinado o híbrido de fabricación y refabricación, donde gestionen su propio flujo inverso y proceso de remanufacturación.

1.4 Políticas de pedidos proporcionales (POUT)

Generalmente, al realizar pedidos de suministros, las empresas, o eslabones de la cadena, tratan de eliminar en su totalidad la diferencia entre el inventario objetivo y el disponible, de manera que se realizan los pedidos necesarios para alcanzar el inventario objetivo. A esta manera de actuar se le conoce como política tradicional de pedidos o *order-up-to (OUT) replenishment policy.*

El inventario objetivo de estas empresas, está definido en gran medida por la demanda que perciben, que recalcamos, en muchas ocasiones difiere de la real. Por lo que utilizar una política de pedidos tradicional puede ocasionar grandes problemas derivados del efecto látigo.

En el artículo Cannella et al. (2021), se describen las *proportional order-up-to (POUT) replenishment policy,* como un modelo de reposición en el cual se emiten pedidos para recuperar parte, pero no la totalidad, de la diferencia entre el inventario objetivo y el disponible. La dimensión de estos pedidos viene definida según unos parámetros de decisión a los que denominaremos *controladores de inventario.* Estas políticas mejoran el rendimiento de las cadenas de suministro tradicionales gracias a su eficacia al hacer frente al efecto látigo, sin embargo, su estudio en las cadenas de suministro de ciclo cerrado es escaso, y este es el objetivo del artículo nombrado:

"Mediante un enfoque de modelización de ecuaciones, demostramos que las políticas de POUT son también un valioso instrumento para mejorar la dinámica de las CLSC. En concreto, descubrimos que el modelo POUT supera a la política tradicional de pedidos en un sistema híbrido de fabricación y refabricación, lo que supone un importante ahorro de costes. Para optimizar el equilibrio clave entre la variabilidad de los pedidos y el inventario, el ajuste de los controladores del inventario debe tener en cuenta no sólo la estructura de costes del CLSC, sino también la tasa media de retorno." (Cannella et al. (2021))

1.5 Objetivo de estudio

Atendiendo a la literatura encontramos que la política POUT se aplicó por primera vez en las CSCC por Tang y Naim (2004). Recientemente, otros trabajos han aplicado esta política para

estudiar el comportamiento de las CSCC, sin embargo, no se ha prestado gran atención al ajuste de los controladores de inventario en estos entornos, a excepción del artículo publicado por Cannella et al. (2021), en el cual se estudia cuáles podrían ser valores recomendados de estos controladores en diferentes escenarios habituales.

El objetivo de este trabajo es adoptar el modelo matemático utilizado por Cannella et al. (2021) y tratar de desarrollar un método de simulación-optimización para mejorar el comportamiento dinámico de estas cadenas mediante el uso de estos controladores.

1.6 Estructura del trabajo

En el Apartado 2 se desarrolla el modelo descrito por Cannella et al. (2021), su funcionamiento, supuestos iniciales, implementación en Python y verificación del correcto funcionamiento del modelo.

En el Apartado 3 estudiaremos que método heurístico implantar, su adaptación y programación en las diferentes variantes en función de los objetivos de estudio. Se comprueba su correcto funcionamiento y se adaptan los parámetros de simulación para optimizar su rendimiento.

En el Apartado 4, ejecutaremos nuestro modelo en diferentes escenarios previamente diseñados y recopilaremos los resultados para su posterior análisis. Con la finalidad de estudiar los comportamientos del modelo y comparar algunos de ellos con los representados en el artículo de referencia.

2 Descripción del modelo

2.1 Aspectos generales

A continuación, en este apartado, se desarrollará el modelo matemático utilizado en el artículo de referencia para este trabajo Cannella et al. (2021). El cual define un sistema híbrido de fabricación y refabricación, es decir, no se define exclusivamente una cadena de suministro de ciclo abierto o cerrado, sino una unión entre ambas, existiendo una función de probabilidad que indica la refabricación o no de un producto.

Implementando correctamente este modelo en un lenguaje de programación, tendremos la posibilidad de simular tantos escenarios como queramos y comprobar cómo reacciona el sistema a cada escenario. En otras palabras, tendremos una herramienta fundamental e imprescindible para alcanzar el objetivo de este trabajo, encontrar los mejores valores posibles (en muchas ocasiones los mejores valores que podamos encontrar no serán óptimos) de los parámetros de control para cada escenario concreto.

2.2 Funcionamiento del modelo

El objetivo de este modelo es simular el funcionamiento de un sistema híbrido de fabricación y refabricación, en un escenario en concreto y durante un número de periodos determinado. De manera que podamos comprobar cómo afecta la variación de los parámetros de control de dicho escenario al funcionamiento o variabilidad del sistema, cuanta menos variabilidad generen en el sistema dichos parámetros de control, mejores serán para el escenario que se está estudiando.

En el artículo de estudio Cannella et al. (2021) y en consonancia con otros trabajos anteriores como Domínguez et al. (2019) se describen tres etapas en cada periodo:

2.2.1 Etapa 1

Al inicio de cada periodo, el inventario disponible aumenta debido a la recepción de productos, tanto de nueva fabricación como refabricados, ver Ecuación 1 y Ecuación 2, contamos por tanto

con un stock inicial, definido por el stock neto del periodo anterior y la recepción de productos, ver Ecuación 3.

$$mc_t = O_{t-Tm-1} \quad \text{(Ecuación 1)}$$

Ecuación 1 Unidades nuevas que finalizan en el periodo t. (Fuente: Cannella et al. (2021)).

Donde O representa las órdenes de producción lanzadas, que se empiezan a fabricar al siguiente periodo del lanzamiento, T_m el tiempo medio de fabricación y por lo tanto O_{t-Tm-1} la orden de producción que se lanzó hace $T_m + 1$ periodos.

$$rc_t = r_{t-Tr-1} \quad \text{(Ecuación 2)}$$

Ecuación 2 Unidades refabricadas que finalizan en el periodo t. (Fuente: Cannella et al. (2021)).

Donde r representa las devoluciones de los clientes, que se empiezan a refabricar en el siguiente periodo de ser recibidas; T_r el tiempo medio de refabricación y por lo tanto r_{t-Tr-1} las devoluciones de clientes que llegaron hace $T_r + 1$ periodos.

$$is_t = ns_{t-1} + mc_t + rc_t \quad \text{(Ecuación 3)}$$

Ecuación 3 Stock inicial en t. (Fuente: Cannella et al. (2021)).

Donde *ns* representa el stock neto, que obtendremos en la Ecuación 6.

2.2.2 Etapa 2

En esta etapa se recibe la demanda de los clientes y se satisface a partir del inventario disponible. La demanda se describe en la Ecuación 4 mediante una variable aleatoria que sigue una distribución normal con media μ y desviación estándar σ, con coeficiente de variación $CV_d = \sigma/\mu$. Por otra parte, en esta etapa se recogen las devoluciones de los clientes, la tasa de retorno viene modelada mediante una variable aleatoria que sigue una distribución normal con media β y desviación estándar ξ, con coeficiente de variación $CV_r = \xi/\beta$, ver Ecuación 5.

$$d_t = max(x_t, 0)//x_t \rightarrow N(\mu, \sigma^2) \quad \text{(Ecuación 4)}$$

Ecuación 4 Demanda del cliente en el periodo t. (Fuente: Cannella et al. (2021)).

$$r_t = y_t d_{t-T_c} // y_t \rightarrow min(max(z_t,0),1) // z_t \rightarrow N(\beta,\xi^2) \quad \text{(Ecuación 5)}$$

Ecuación 5 Unidades refabricadas que finalizan en t. (Fuente: Cannella et al. (2021)).

A medida que se satisface la demanda, el inventario disponible disminuye, denominamos stock neto al stock disponible al final de cada periodo, ver Ecuación 6. Cuando este es positivo, aún tenemos inventario para seguir satisfaciendo la futura demanda, mientras que si es negativo no hemos podido satisfacer la demanda del periodo por falta de existencias y se tendrá que satisfacer cuando tengamos inventario disponible.

$$ns_t = is_t - d_t \quad \text{(Ecuación 6)}$$

Ecuación 6 Stock neto en t. (Fuente: Cannella et al. (2021)).

El último factor determinante de esta etapa es el trabajo en curso al final de "t", hace referencia a los productos que se encuentran en las líneas de fabricación o refabricación y que, por lo tanto, aún no forman parte del inventario disponible. Ver Ecuación 7.

$$w_t = w_{t-1} + (o_{t-1} - mct) + (r_{t-1} - rc_t) \quad \text{(Ecuación 7)}$$

Ecuación 7 Trabajo en curso en t. (Fuente: Cannella et al. (2021)).

2.2.3 Etapa 3

Finalmente, en la etapa de abastecimiento se generan nuevas órdenes de producción de productos nuevos, de manera que se pueda satisfacer la demanda que no alcanzamos a satisfacer con productos remanufacturados. Hasta ahora no ha aparecido ningún parámetro de control en las ecuaciones del modelo, sin embargo, en esta última etapa se aplican las políticas de POUT explicadas al inicio del trabajo, lo que supone que se van a tomar decisiones sobre algunos valores del modelo por parte del director de operaciones. Es importante analizar que valores dar a estos parámetros de control ya que afectan directamente al funcionamiento del sistema, y este es el objeto de estudio del trabajo que se desarrolla.

Con la Ecuación 8 se calculan las órdenes de producción de cada periodo, donde *"T_i y T_w son parámetros de decisión. que pueden denominarse como la constante de tiempo de los controladores de inventario. Definen la parte de la diferencia entre el objetivo y el inventario real (T_i para el inventario disponible y T_w para el trabajo en curso) que debe tener en cuenta la regla de reposición." (Cannella et al. (2021))*. $\widehat{d_t}$ es la previsión de la demanda, calculada con la Ecuación 9, donde α es el tercer parámetro de control, sirve para determinar la sensibilidad que debe tener la regla de previsión de demanda a las variaciones de esta, suaviza por tanto el efecto látigo.

$$o_t = max((\widehat{d_t} - rc_t) + \frac{1}{T_i}(ss_t - ns_t) + \frac{1}{T_w}(tw_t - w_t), 0) \quad \text{(Ecuación 8)}$$

Ecuación 8 Orden de producción en el periodo t. (Fuente: Cannella et al. (2021)).

$$\widehat{d_t} = \alpha d_t + (1-\alpha)\widehat{d_{t-1}} \quad \text{(Ecuación 9)}$$

Ecuación 9 Previsión de la demanda del periodo t. (Fuente: Cannella et al. (2021)).

En el cálculo de las órdenes de producción participan también el stock de seguridad y el trabajo en curso objetivo, valores calculados según la Ecuación 10 y Ecuación 11, donde encontramos los dos últimos parámetros de decisión ε y T_p. ε representa el nivel de protección contra falta de existencias, por lo que cuanto mayor sea, mayor será el stock de seguridad. T_p representa la estimación del plazo de entrega, en el artículo de estudio se calcula como una estimación siguiendo la propuesta de Tang y Naim (2004), es decir, realizando una media ponderada en función de β como podemos ver en la Ecuación 12, parte de este trabajo tratará de comparar los valores resultantes de esta estimación y de una búsqueda heurística.

$$ss_t = \varepsilon \widehat{d_t} \quad \text{(Ecuación 10)}$$

Ecuación 10 Stock de seguridad del periodo t. (Fuente: Cannella et al. (2021)).

$$tw_t = T_p \widehat{d_t} \quad \text{(Ecuación 11)}$$

Ecuación 11 Trabajo en curso objetivo del periodo t. (Fuente: Cannella et al. (2021)).

$$T_p = (1-\beta)T_m + \beta T_r \quad \text{(Ecuación 12)}$$

Ecuación 12 Estimación del plazo de entrega. (Fuente: Cannella et al. (2021)).

2.3 Métricas de rendimiento

Mediante el ratio de variabilidad de pedidos *OVR* medimos la fluidez de la producción, como vemos en la Ecuación 13, comparamos la varianza de los pedidos de producción con la varianza de la demanda. Por otro lado, el rendimiento de los inventarios se mide utilizando el ratio de variabilidad

de inventarios *IVR*, Ecuación 14.

$$OVR = \frac{var(o_t)}{var(d_t)} \quad \text{(Ecuación 13)}$$

Ecuación 13 Ratio de variabilidad de pedidos. (Fuente: Cannella et al. (2021)).

$$IVR = \frac{var(ns_t)}{var(d_t)} \quad \text{(Ecuación 14)}$$

Ecuación 14 Ratio de variabilidad de inventario. (Fuente: Cannella et al. (2021)).

Para poder estudiar simultáneamente ambos aspectos se define una métrica unificada a la que denominaremos como *J*, representada en la Ecuación 15, donde k_o y k_i indican los pesos de cada término, es decir, k_o será mayor que k_i en un escenario donde tenga mayor importancia la fluidez de la producción que el rendimiento de los inventarios y viceversa (k_o+ k_i=1). Minimizar *J* será por tanto la función objetivo durante el desarrollo de todo el trabajo.

$$J = k_o\sqrt{OVR} + k_i\sqrt{IVR} \quad \text{(Ecuación 15)}$$

Ecuación 15 Métrica de variabilidad total. (Fuente: Cannella et al. (2021)).

2.4 Definición de parámetros

- **Parámetros de control del modelo:** $T_i, T_w, \alpha, T_p, \varepsilon$**.**

  Son los valores que vamos a poder ajustar con el fin de optimizar el funcionamiento de la cadena de suministro. Como veremos más adelante, comúnmente en la literatura algunos de ellos son adoptados como factores fijos, a fin de simplificar el diseño experimental cuando no son objeto del estudio en cuestión.

- **Parámetros incontrolables fijos del modelo:** T_m, T_r, T_c, μ, CV_d**.**

  Son aquellos que nunca van a variar, indiferentemente del escenario que se simule.

- **Parámetros incontrolables variables del modelo:** β, CV_r, k_i, k_o,(k_s ver apartado 3.1)

  Son aquellos factores cuyos valores no podemos controlar y varían debido a causas externas al experimento. Como puede ser la tasa de retorno de productos o el interés que tenga la empresa en la variabilidad del inventario en cierta época del año.

- **Parámetros del experimento:** T, E.

  Estos parámetros se utilizan para controlar el número de veces que ejecutamos el modelo en nuestro experimento, así como el número de veces que repetimos el experimento antes de analizar los resultados.

Los parámetros del experimento, se mantendrán fijos durante todo el trabajo, por tanto, ya podemos fijar los siguientes valores:

- $T_m = 4, T_r = 2, Tc = 16, \mu = 100, CV_d = 0.3$

- T: Número de periodos en cada simulación, se establece en 2100 al igual que en el artículo de referencia, para el cálculo de los resultados no se tendrán en cuenta los 100 primeros periodos ya que se consideran como un horizonte de calentamiento.

- E: Número de repeticiones del experimento, se repite 10 veces, desde e=1 hasta e<11

2.5 Aplicación y validación del modelo

Una vez desarrollado el funcionamiento del modelo, debemos implementarlo en un lenguaje de programación, en este caso lo haremos con Python. Python es un lenguaje de programación interpretado de alto nivel, de propósito general, que contiene numerosas bibliotecas para manejar funciones matemáticas y científicas. Entre sus desventajas destacan la lentitud y el consumo de memoria, esto se debe a su naturaleza dinámica y versatilidad, siendo las ventajas fundamentales que hacen elegir este lenguaje.

Para comprobar el correcto funcionamiento del modelo programado (Anexo I), simulamos escenarios similares a los representados en Cannella et al. (2021) y comprobamos que el comportamiento de los resultados obtenidos en nuestro modelo es similar al representado en dicho artículo.

Para ello debemos configurar el modelo al igual que en el artículo, de manera que:

- ε actúe como factor fijo de valor 1

- T_p actúe como factor fijo de valor $T_p = (1-\beta)T_m + \beta T_r$

- $T_i = T_w$ (Esta es una medida que se aplica en numerosos estudios de la literatura con el objetivo de simplificar el modelo)

El resto de factores y parámetros de control ($\beta, \alpha, k_i, k_o, CV_r$) variarán en cada comprobación en función del escenario que se simule. A continuación se muestran algunos escenarios estudiados en Cannella et al. (2021) y los resultados obtenidos con nuestro modelo para dichos escenarios.

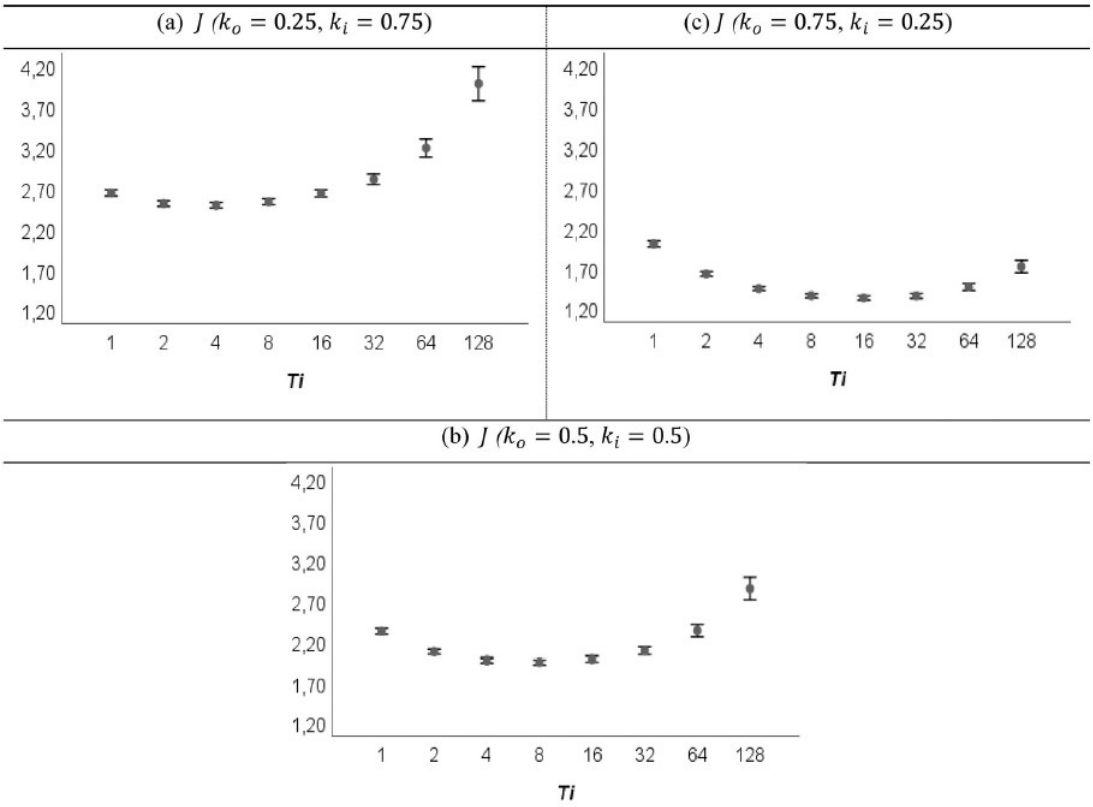

Figura 2.1 Valores de *J* en función de *Ko* y *Ki* (Fuente: Cannella et al., 2021).

Realizamos tres primeras comprobaciones, una en cada caso (nótese que en estas primeras comprobaciones los valores de α y β son indiferentes).

- Para el caso A con T_i = 4 obtenemos en la simulación un valor de J = 2,448
- Para el caso B con T_i = 8 obtenemos en la simulación un valor de J =1,929
- Para el caso C con T_i = 16 obtenemos en la simulación un valor de J =1,291

En las siguientes cuatro comprobaciones verificaremos la correcta reacción del modelo ante la variación del valor de β. Para ello nos fijaremos en la siguiente imagen.

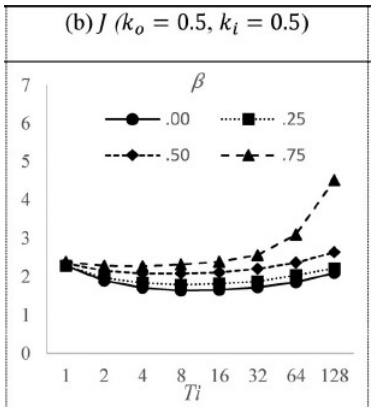

Figura 2.2 Valores de *J* en función de *Ko*, *Ki* y β (Fuente: Cannella et al., 2021).

Fijamos T_i = 4 y comprobamos que los valores de J ascienden a medida que asciende el valor de β.

- Para β = 0 obtenemos en la simulación un valor de J = 1,953
- Para β = 0.25 obtenemos en la simulación un valor de J = 1,965
- Para β = 0.5 obtenemos en la simulación un valor de J = 2,015
- Para β = 0.75 obtenemos en la simulación un valor de J = 2,075

Por último, comprobamos la reacción del modelo ante la variación de α, para ello nos fijamos en la siguiente gráfica y comprobamos cuatro valores, de manera que podamos verificar que para el caso de k_o= 0.5 y α = 0, el valor de J aumenta conforme lo hacen T_i y T_w, mientras que para otros valores de α el sistema se comporta de forma diferente.

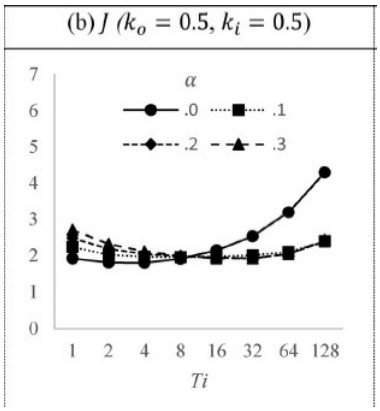

Figura 2.3 Valores de *J* en función de *Ko*, *Ki* y α (Fuente: Cannella et al., 2021).

- Para α = 0 y T_i = 2 obtenemos en la simulación un valor de J = 1,437
- Para α = 0 y T_i = 32 obtenemos en la simulación un valor de J = 2,310

- Para α = 0.3 y T_i = 2 obtenemos en la simulación un valor de J = 2,302
- Para α = 0.3 y T_i = 32 obtenemos en la simulación un valor de J = 1,555

Tras las comprobaciones realizadas podemos afirmar que hemos implementado correctamente en Python el modelo de simulación de un sistema híbrido de fabricación y refabricación descrito inicialmente.

3 Metaheurística

En el capítulo anterior implementamos el modelo en Python, el siguiente paso será por tanto desarrollar un algoritmo heurístico para analizar qué valores de los parámetros de decisión benefician más a la cadena de suministro en cada escenario. Denominamos escenario a la situación en la que cada parámetro incontrolable (ver punto 2.4) toma un valor concreto, no existen por tanto dos escenarios iguales.

Con este fin trabajaremos con el algoritmo *Iterated greedy*. Este es un algoritmo heurístico que se basa en la destrucción parcial de una solución inicial y la posterior reconstrucción de esta, dando lugar a otra solución completa, distinta de la inicial.

En el grado de Ingeniería de Organización Industrial aplicamos esta metaheurística a problemas de programación y control de la producción, donde estas soluciones vienen determinadas por vectores de secuencias de trabajos y nuestro objetivo es secuenciar dichos trabajos de manera que alcancemos el mejor valor posible de cierta función objetivo.

Sin embargo, en el problema que nos acontece, no tendría ningún sentido alterar las soluciones cambiando el orden de los valores de la solución (parámetros de decisión), ya que estos valores son totalmente independientes entre sí, aunque afecten de forma conjunta al valor de la función objetivo. En este sentido, se describe a continuación como vamos a implementar este algoritmo.

Inicialmente la idea del proyecto era permitir al algoritmo realizar combinaciones con total libertad sobre todas las variables o parámetros de control, sin embargo, tras la realización de numerosas pruebas y simulaciones de diferentes escenarios, se tomó la decisión de simplificar el modelo adoptando la hipótesis de igualar T_i y T_w, al igual que se realiza en Cannella et al. (2021) como vimos en el apartado 2.5.

Esta medida se aplica con el fin de reducir el abanico de soluciones posibles a estudiar por el algoritmo, ya que en otro caso los tiempos de simulación necesarios para alcanzar cierta convergencia en las soluciones serían totalmente desproporcionados.

3.1 Adaptación del modelo

Como adelantamos en el apartado 2.4, los parámetros de control se utilizan para poder dirigir el funcionamiento de la cadena de suministro hacia el comportamiento que más nos interese (como por ejemplo incrementar el nivel de existencias, aunque esto suponga un aumento en el coste de inventarios o viceversa). También mencionamos en el referido apartado 2.4 que comúnmente son considerados en la literatura como factores fijos, esto es lo que ocurre en el estudio realizado por Cannella et al. (2021):

- ε=1: Se ha comprobado que puede ser una decisión razonable cuando el nivel de demanda y los rendimientos son inciertos y deben evitarse los agotamientos de existencias. véase, por ejemplo, Cannella, Bruccoleri y Framinan (2016).

- $T_p = (1-\beta)T_m + \beta T_r$ (ver Ecuación 12): Sugerencia de Tang y Naim (2004) de manera que el rendimiento del inventario no se vea penalizado a causa de la deriva del mismo.

Como consecuencia de esta común adopción de valores, estos parámetros no han sido optimizados en conjunto con otros parámetros de control, como los controladores proporcionales, y teniendo en cuenta toda la dinámica de la cadena de suministro.

Este es el principal objetivo del trabajo en cuestión, y para poder llevarlo a cabo debemos permitir que el modelo heurístico que implementaremos más adelante, tenga poder de decisión sobre los valores de estos parámetros.

Como podemos observar a través de las Ecuación 8, Ecuación 10 y Ecuación 11, estos parámetros de control influyen directamente sobre las ordenes de producción, y por tanto sobre el stock neto. En el artículo de referencia, el nivel de este stock neto se mantiene bajo control gracias a la consideración que acabamos de explicar, la adopción de ε y T_p como factores fijos. Sin embargo, si vamos a permitir que la heurística actúe sobre estos valores, utilizando el código implementado en el punto 2.5 como función para evaluar cada solución candidata aportada por la heurística, es necesario adaptar el código de forma que se contemple este nivel de stock, así como el porcentaje de satisfacción de la demanda.

Uno de los objetivos al optimizar la dinámica de esta cadena de suministro es minimizar el stock neto medio de cada simulación, pero cumpliendo un porcentaje mínimo de satisfacción de la demanda. Para ello aplicamos las siguientes medidas:

1. En cada simulación (recordamos que cada experimento se simula 10 veces con 2100 periodos) comprobamos que el stock neto ha sido positivo más del 90 % de las veces, en caso contrario no se calculan las métricas de rendimiento y dicha simulación queda invalidada. Si más de una simulación en un experimento queda invalidada, todo ese experimento es inválido y declaramos que esa combinación de valores de los parámetros de control para el escenario en concreto no conlleva una solución factible.

2. En la heurística utilizaremos la función de coste del modelo (Ecuación 15) como función objetivo de la misma, la cual trataremos de minimizar incluyendo el coste del nivel de inventario:

$$Jmod = k_o\sqrt{OVR} + k_i\sqrt{IVR} + k_s(\frac{nsm}{100}) \qquad \text{(Ecuación 16)}$$

Ecuación 16: Función de coste adaptada. (Fuente: Elaboración propia).

Siendo k_s el peso que asignamos al nivel de stock neto medio $k_s = 1 - k_o - k_i$. Y *nsm* el stock neto medio de cada simulación, calculado como la media de todos los stock neto de cada periodo en una simulación.

El código de la adaptación se recoge en el Anexo I

3.2 Adaptación *"Iterated greedy"*

Aunque la esencia del algoritmo siempre va a ser la misma, se realizan dos variantes. En primer lugar, optimizaremos la dinámica de la cadena de suministro mediante 3 controladores de inventario, estableciendo los 2 restantes como factores fijos, ya que se trata de un problema más sencillo. Y posteriormente implementaremos una variante más compleja en la que optimizaremos la cadena teniendo en cuenta todos los controladores simultáneamente.

Como adelantamos en el apartado 3.1, durante todo el proceso de búsqueda el objetivo será minimizar *Jmod*. Es decir, en un proceso de búsqueda convencional, en cada iteración calcularíamos el resultado de una función matemática o similar y este sería el valor que debemos maximizar o minimizar, sin embargo, en este caso, en cada búsqueda ejecutaremos una función, que contendrá el código implementado en el apartado 3.1, el cual nos devolverá un valor de *Jmod*, que será el que buscaremos minimizar.

Cabe destacar que, tras la modificación realizada sobre el modelo original en el apartado 3.1, no será posible comparar los valores que obtengamos de la función objetivo con los del artículo de referencia ya que *Jmod* no es comparable con *J*. Sin embargo, podemos encontrar una aproximación a *J* repartiendo a partes iguales el valor de K_s entre K_o y K_i, cuanto menor sea por tanto el valor de K_s más fiable será la aproximación, representada en las tablas de resultados como *Japrox*.

3.2.1 Primera variante

Consideramos como parámetros a definir únicamente T_i, T_w y T_p, por lo que las soluciones estarían formadas por vectores definidos con la siguiente secuencia fija e invariable:

Solución = $(T_i = T_w, T_p)$

1. Dada una solución inicial aleatoria, iniciamos un bucle donde alteraremos esta solución hasta que la función objetivo no mejore x veces.

2. En este bucle, aleatoriamente seleccionamos el parámetro que vamos a eliminar, y posteriormente lo sustituimos por cada uno de los valores posibles para este parámetro, quedándonos con la mejor solución.

Por ejemplo, se da el caso de que aleatoriamente se destruye $T_i = T_w$, iterativamente (en un segundo bucle anidado al inicial) se estudia cual sería el mejor valor de los 64 posibles (*Best improvement*) y se insertaría en su posición. Si el valor de $T_i = T_w$ insertado es diferente al que fue destruido significa que se ha encontrado un valor que mejora la función objetivo.

3. Volvemos a escoger aleatoriamente un parámetro a destruir diferente al inmediatamente anterior y realizamos el mismo proceso. (Inicialmente el modelo podía escoger aleatoriamente entre 2 parámetros, sin embargo al realizar la simplificación $T_i = T_w$ solo queda un parámetro diferente al que acaba de ser modificado.)

4. La búsqueda finaliza cuando el valor de la función objetivo no mejore x veces.

Dicho código queda representado en el Anexo I

3.2.2 Segunda variante

Es una variante más compleja, ya que en este caso, se va a optimizar la cadena teniendo en cuenta simultáneamente todos los parámetros de control. Para ello, llevamos a cabo un procedimiento similar al de la primera variante, pero en este caso se escogen aleatoriamente dos de los cuatro parámetros posibles, y se incorpora un tercer bucle para estudiar todo el abanico de búsqueda existente para cada pareja de parámetros.

Es importante señalar que al igual que en la primera variante nos asegurábamos de que en cada iteración del bucle principal se estudiaba una variable diferente, en esta segunda variante es indispensable que la pareja de parámetros a estudiar sea diferente a la de la iteración anterior.

Por último, destacar que se deben tener en cuenta todas las parejas de parámetros posibles (6 variantes) y programar los bucles de búsqueda correspondientes para cada pareja.

El vector solución quedará definido tal que:

Solución =$(T_i = T_w, T_p, \alpha, \varepsilon)$

El código de esta variante se representa en el Anexo I

3.2.3 Detalles de la búsqueda

La búsqueda iterativa de los valores para los diferentes parámetros se realiza de distinta forma para cada uno de ellos en función de su naturaleza. Recogemos los valores máximo y mínimo para cada uno de estos parámetros, así como el número de unidades en las que variarán en cada iteración:

- T_i: Constante de tiempo asociada a los inventarios. Toma valores entre 0 y 64. Al variar su valor en cada iteración dentro de un bucle, se hace inicialmente de cinco en cinco unidades (+/- 5). Una vez encontrado el mejor valor entre los posibles, se realiza una segunda búsqueda desde este valor encontrado menos 10 unidades, hasta el valor encontrado más 10 unidades, con un incremento de una unidad en cada iteración.

- T_w: Constante de tiempo asociada al trabajo. Toma valores entre 0 y 64. Al variar su valor en cada iteración dentro de un bucle, se hace inicialmente de cinco en cinco unidades (+/- 5). Una vez encontrado el mejor valor entre los posibles, se realiza una segunda búsqueda desde este valor encontrado menos 10 unidades, hasta el valor encontrado más 10 unidades, con un incremento de una unidad en cada iteración.

- T_p: Estimación del plazo de entrega. Toma valores entre 0.5 y 6. Al variar su valor en cada iteración dentro de un bucle, se hace en 0.25 unidades (+/- 0.25)

- α: Factor de suavización. Toma valores entre 0 y 1. Al variar su valor en cada iteración dentro de un bucle, se hace en 0.1 unidades (+/- 0.1)

- ε: Nivel de protección contra falta de existencias. Toma valores entre 0 y 3. Al variar su valor en cada iteración dentro de un bucle, se hace en 0.1 unidades (+/- 0.1).

Cabe destacar, que la complejidad añadida en la segunda variante, se debe fundamentalmente a la incorporación de un tercer bucle, necesario para poder estudiar todo el abanico de búsqueda que encontramos en cada pareja de parámetros. Es decir, en la primera variante al eliminar por ejemplo $T_i = T_w$, el abanico de búsqueda era desde 0 hasta 64 con un intervalo inicial de 5, lo que se traduce en 64/5=12 opciones aproximadamente. En el caso de la segunda variante, si se elimina por ejemplo $T_i = T_w$ y T_p, debemos estudiar $12 \cdot (\frac{6-0.5}{0.25} + 1)$=276 opciones.

3.3 Verificación del modelo heurístico

Para comprobar el funcionamiento del modelo diseñado se simulan diferentes escenarios característicos, en los que se atribuyen unos pesos (k_o, k_i y k_s) exageradamente desbalanceados, de manera que se pueda visualizar con mayor facilidad la variación de los resultados en función de dichos parámetros.

Con respecto a la duración del proceso de optimización, es decir, el número de **no** mejoras, se calcula a base de prueba y error. Para la variante 1 se comenzó con un valor de 100, con el cual ya se percibía cierta convergencia en los resultados, por lo tanto, se ascendió a 300 ya que no suponía un aumento considerable en la duración de los experimentos, con el fin de ajustar aún más los resultados obtenidos.

En la siguiente Tabla 3.1 podemos visualizar los resultados obtenidos y como convergen a diferentes valores en función de los pesos asignados a la fluidez de la producción k_o, el rendimiento de los inventarios k_i y el stock neto medio k_s.

Para esta primera prueba se fijaron los valores de $\alpha = 0.1, \beta = 0.5$ y $\varepsilon = 1$.

Tabla 3.1 Verificación variante 1 (Fuente: elaboración propia).

K_o	K_i	Nº NO mejora	$T_i = T_w$	T_p	J mod	J aprox
0.8	0.1	300	56	4	0,772	0,837
0.8	0.1	300	36	3,75	0,745	0,817
0.8	0.1	300	41	3,75	0,744	0,817
0.8	0.1	300	46	3,75	0,749	0,838
0.8	0.1	300	46	3,75	0,743	0,824
0.8	0.1	300	46	3,75	0,749	0,838
0.8	0.1	300	36	3,75	0,749	0,821
0.8	0.1	300	41	3,75	0,743	0,824
0.8	0.1	300	41	3,75	0,747	0,825
0.8	0.1	300	46	3,75	0,743	0,822
0.1	0.8	300	3	3,75	2,317	2,475
0.1	0.8	300	2	3,75	2,32	2,413
0.1	0.8	300	2	3,75	2,323	2,457
0.1	0.8	300	3	3,75	2,327	2,461
0.1	0.8	300	2	3,75	2,319	2,431
0.1	0.8	300	2	3,75	2,319	2,463
0.1	0.8	300	2	3,75	2,321	2,403
0.1	0.8	300	3	3,75	2,337	2,506
0.1	0.8	300	2	3,75	2,324	2,41
0.1	0.8	300	2	3,75	2,321	2,458
0.1	0.1	300	61	4	1,471	1,89
0.1	0.1	300	51	3,75	1,294	1,809
0.1	0.1	300	61	4	1,47	1,864
0.1	0.1	300	61	4	1,466	1,856
0.1	0.1	300	41	3,75	1,305	1,74
0.1	0.1	300	51	3,75	1,294	1,768
0.1	0.1	300	61	4	1,468	1,834
0.1	0.1	300	63	4	1,469	1,867
0.1	0.1	300	26	3,627	1,224	1,711
0.1	0.1	300	61	4	1,465	1,864

Por otro lado, en la segunda variante a diferencia de la primera, la convergencia de los resultados es menos notable con un valor de **no** mejora de 60. Por ello, para comprobar un funcionamiento relevante del sistema, se decide ir incrementando este valor en vez de modificar los pesos como se hizo en el primer caso. De esta forma, además de comprobar el correcto desempeño de la heurística, podremos determinar un número de no mejora razonable, que permita una convergencia de los resultados, pero sin unos tiempos de experimentación demasiado altos.

Esta diferencia de convergencia se debe a la existencia de muchas más combinaciones de valores ya que entran en juego dos nuevos parámetros. Además, la duración de los experimentos aumenta considerablemente a medida que se incrementa el número de **no** mejora.

A pesar de ello, en la Tabla 3.2 podemos notar ciertos comportamientos del modelo que señalan un correcto funcionamiento de este, como pueden ser el aumento de la convergencia de $T_i = T_w$ o α a medida que aumenta la duración de los experimentos, o ciertas relaciones de proporcionalidad entre T_p y ε que se detallarán en el próximo capítulo.

Tabla 3.2 Verificación variante 2 (Fuente: elaboración propia).

K_o	K_i	Nº NO mejora	$T_i = T_w$	T_p	α	ε	J mod	J aprox
0.8	0.1	60	31	3	0,1	1,2	0,91	0,985
0.8	0.1	60	51	3,75	0,147	0,5	0,915	0,993
0.8	0.1	60	38	1,5	0,1	2,8	0,909	0,999
0.8	0.1	60	46	2	0,1	2,3	0,91	0,992
0.8	0.1	60	36	3,25	0,1	1	0,911	0,991
0.8	0.1	60	44	2,5	0,1	1,8	0,915	0,983
0.8	0.1	60	41	2	0,1	2,3	0,915	0,991
0.8	0.1	60	36	2,25	0,1	2	0,913	0,996
0.8	0.1	60	36	2,25	0,1	2	0,913	0,996
0.8	0.1	60	54	3	0,19	1,2	0,919	1,007
0.8	0.1	90	34	3,75	0,1	0,5	0,915	0,982
0.8	0.1	90	56	2	0,2	2,2	0,924	1,009
0.8	0.1	90	41	3,25	0,1	1	0,91	0,999
0.8	0.1	90	43	4	0,1	0,3	0,912	0,996
0.8	0.1	90	47	2,25	0,1	2	0,907	0,997
0.8	0.1	90	36	3,25	0,1	1	0,909	0,994
0.8	0.1	90	43	2,5	0,1	1,8	0,908	0,987
0.8	0.1	90	33	3,25	0,1	1	0,911	1,004
0.8	0.1	90	41	4,25	0,1	0	0,909	0,982
0.8	0.1	90	46	2	0,1	2,3	0,911	1,001
0.8	0.1	150	46	1,75	0,1	2,5	0,905	0,985
0.8	0.1	150	41	2,75	0,1	1,5	0,9	0,99
0.8	0.1	150	41	2,25	0,1	2	0,91	0,994
0.8	0.1	150	41	3,25	0,1	1	0,907	0,984
0.8	0.1	150	41	2,75	0,1	1,5	0,906	1
0.8	0.1	150	41	1,5	0,1	2,8	0,914	0,996
0.8	0.1	150	41	3,75	0,1	0,5	0,905	0,984
0.8	0.1	150	31	2,5	0,1	1,7	0,911	0,999
0.8	0.1	150	46	1,5	0,1	2,8	0,908	1
0.8	0.1	150	47	3,5	0,1	0,8	0,911	0,987

Con el fin de facilitar la comprensión de los resultados mostrados, se detallan en función de la media y la desviación estándar de los valores obtenidos para cada parámetro.

Tabla 3.3 Datos $T_i = T_w$ verificación variante 2 (Fuente: elaboración propia).

Nº NO mejora	**Parámetro**	**Media**	**Desviación. E**
60	$T_i = T_w$	41,3	7,349
90	$T_i = T_w$	42	6,848
150	$T_i = T_w$	41,6	4,502

Tabla 3.4 Datos T_p verificación variante 2 (Fuente: elaboración propia).

Nº NO mejora	Parámetro	Media	Desviación. E
60	T_p	2,550	0,685
90	T_p	3,050	0,823
150	T_p	2,550	0,806

Tabla 3.5 Datos α verificación variante 2 (Fuente: elaboración propia).

Nº NO mejora	Parámetro	Media	Desviación. E
60	α	0,114	0,031
90	α	0,110	0,032
150	α	0,100	0,000

Tabla 3.6 Datos ε verificación variante 2 (Fuente: elaboración propia).

Nº NO mejora	Parámetro	Media	Desviación. E
60	ε	1,710	0,711
90	ε	1,210	0,821
150	ε	1,710	0,814

Tras estas pruebas se decide establecer a **90** el número de **no mejora** ya que se considera suficiente para el estudio del comportamiento del sistema en el trabajo que nos atañe. Al aumentar aún más este valor se consigue mayor convergencia en los valores de $T_i = T_w$ y α, sin embargo los tiempos de simulación aumentan drásticamente.

Una vez definidos estos parámetros y comprobado que los modelos convergen hacia ciertos resultados, debemos comprobar que realmente mejoramos los valores obtenidos en el artículo de referencia. Esto no es algo trivial ya que en Cannella et al. (2021) se analizan las interacciones de los parámetros dos a dos, pero no contamos con simulaciones de escenarios concretos con los que poder equiparar. Adicionalmente, como se especificó en un capitulo anterior, no podemos comparar directamente Jmod, sino que necesitamos apoyarnos en una aproximación Japrox que será más fiable cuanto menor sea k_s. Pero a su vez, no podemos reducir drásticamente el valor de k_s, porque estaríamos comparando con una situación irreal en la que no se tiene en cuenta el stock neto almacenado.

Para llevar a cabo este proceso nos fijaremos en la figura 2.2, concretamente vamos a escoger el caso en el que $\beta = 0.25$ y vamos a simular 4 escenarios diferentes por cada variante que se pueden asemejar al de la referida figura, con el fin de observar que valores obtenemos de Japrox.

Los datos resultantes de la variante 1 se recogen en la siguiente tabla 3.7:

Tabla 3.7 Resultados simulación variante 1, β = 0.25 (Fuente: elaboración propia).

K_o	K_i	K_s	α	$T_i = T_w$	T_p	J aprox
0.33	0.33	0.33	0,1	26	3,75	1,69
0.33	0.33	0.33	0,3	46	3,75	1,63
0.4	0.4	0.2	0,1	31	3,75	1,72
0.4	0.4	0.2	0,3	51	3,75	1,6

Al cotejar estos valores con los de la figura 2.2 observamos que, los valores de la función objetivo obtenidos por nuestra heurística son ligeramente mejores que los obtenidos en el modelo de Cannella et al. (2021). Sin embargo, resalta a la vista que los valores de $T_i = T_w$ óptimos distan bastante de un modelo a otro, esto puede ocurrir debido a la libertad que tiene nuestra heurística sobre el valor de T_p, a causa de la adición de un nuevo peso al cálculo del valor de la función objetivo, o una combinación de ambos factores. Se estudiará más en detalle el comportamiento de los diferentes parámetros en el siguiente capítulo.

Por su parte, la segunda variante nos ofrece los siguientes resultados representados en la tabla 3.8:

Tabla 3.8 Resultados simulación variante 2, β = 0.25 (Fuente: elaboración propia).

K_o	K_i	K_s	$T_i = T_w$	T_p	α	ε	J aprox
0.33	0.33	0.33	51	2,25	0,3	2,3	1,59
0.37	0.37	0.26	51	3,5	0,5	1	1,61
0.4	0.4	0.2	57	4,25	0,4	0,3	1,62
0.45	0.45	0.1	46	2	0,4	2,6	1,6

En ellos podemos apreciar que esta variante mejora ligeramente los valores de la función objetivo con respecto a la primera variante y, en consecuencia, también con respecto al artículo de referencia, pero como ya hemos comentado, a costa de tiempos de simulación muy elevados.

Por si fuera de interés, se adjunta en el Anexo II los códigos de las variantes iniciales donde no se considera la igualdad entre T_i y T_w, así como las tablas de los resultados obtenidos.

4 Diseño de experimento y resultados

El presente experimento tiene como objetivo establecer ciertas hipótesis o teorías sobre relaciones causales entre variables, evaluar la eficacia del modelo de optimización aplicado, así como comparar los valores de parámetros preestablecidos con los propuestos por el modelo.

En este trabajo nos limitaremos a plasmar el comportamiento del modelo en distintos escenarios, tratando de aportar sugerencias para la configuración de este tipo de cadenas de suministro, pero sin buscar la razón última de estos comportamientos, ya que nos encontramos ante un modelo dinámico cuyo funcionamiento no podemos comprender directamente sin un análisis detallado del mismo.

Con este fin, se simulan ambas variantes 10 veces en 9 escenarios diferentes, definidos por las combinaciones de los siguientes valores:

- K_o=0.1 o 0.8
- K_i=0.1 o 0.8
- K_s=0.1 o 0.8
- β=0.25, 0.5 o 0.75

Sabiendo que: $K_o + K_i + K_s = 1$

Con el fin de simplificar las tablas de resultados en ocasiones se denotará:

- **A** al caso en el que k_o=0.8, k_i=0.1 y k_s=0.1
- **B** al caso en el que k_o=0.1, k_i=0.8 y k_s=0.1
- **C** al caso en el que k_o=0.1, k_i=0.1 y k_s=0.8

4.1 Resultados variante 1

Del mismo modo que en el capítulo anterior, para la primera variante fijaremos los valores de α = 0,1 y ε = 1. Los resultados obtenidos con la primera variante se recogen en las siguientes tablas, acompañadas por la media y desviación estándar de cada uno de los parámetros en cada escenario.

Tabla 4.1 Variante 1, caso β=0.25 (Fuente: elaboración propia).

K_o	K_i	Nº NO mejora	$T_i = T_w$	T_p	J mod	J aprox
0.8	0.1	300	56	4	0,772	0,837
0.8	0.1	300	36	3,75	0,745	0,817
0.8	0.1	300	41	3,75	0,744	0,817
0.8	0.1	300	46	3,75	0,749	0,838
0.8	0.1	300	46	3,75	0,743	0,824
0.8	0.1	300	46	3,75	0,749	0,838
0.8	0.1	300	36	3,75	0,749	0,821
0.8	0.1	300	41	3,75	0,743	0,824
0.8	0.1	300	41	3,75	0,747	0,825
0.8	0.1	300	46	3,75	0,743	0,822
0.1	0.8	300	3	3,75	2,317	2,475
0.1	0.8	300	2	3,75	2,32	2,413
0.1	0.8	300	2	3,75	2,323	2,457
0.1	0.8	300	3	3,75	2,327	2,461
0.1	0.8	300	2	3,75	2,319	2,431
0.1	0.8	300	2	3,75	2,319	2,463
0.1	0.8	300	2	3,75	2,321	2,403
0.1	0.8	300	3	3,75	2,337	2,506
0.1	0.8	300	2	3,75	2,324	2,41
0.1	0.8	300	2	3,75	2,321	2,458
0.1	0.1	300	61	4	1,471	1,89
0.1	0.1	300	51	3,75	1,294	1,809
0.1	0.1	300	61	4	1,47	1,864
0.1	0.1	300	61	4	1,466	1,856
0.1	0.1	300	41	3,75	1,305	1,74
0.1	0.1	300	51	3,75	1,294	1,768
0.1	0.1	300	61	4	1,468	1,834
0.1	0.1	300	63	4	1,469	1,867
0.1	0.1	300	26	3,627	1,224	1,711
0.1	0.1	300	61	4	1,465	1,864

Tabla 4.2 Variante 1A caso β=0.25 (Fuente: elaboración propia).

Parámetro	Media	Desviación. E
$T_i = T_w$	43,50	5,892
T_p	3,775	0,079

Tabla 4.3 Variante 1B caso β=0.25 (Fuente: elaboración propia).

Parámetro	Media	Desviación. E
$T_i = T_w$	2,3	0,483
T_p	3,75	0

Tabla 4.4 Variante 1C caso β=0.25 (Fuente: elaboración propia).

Parámetro	Media	Desviación. E
$T_i = T_w$	53,7	12
T_p	3,888	0,149

Tabla 4.5 Variante 1, caso β=0.5 (Fuente: elaboración propia).

K_o	K_i	Nº NO mejora	$T_i = T_w$	T_p	J mod	J aprox
0.8	0.1	300	41	3,25	0,909	0,991
0.8	0.1	300	41	3,25	0,91	0,976
0.8	0.1	300	36	3,25	0,905	0,987
0.8	0.1	300	56	3,5	0,929	1,012
0.8	0.1	300	51	3,5	0,931	1,002
0.8	0.1	300	46	3,5	0,924	1,006
0.8	0.1	300	56	3,5	0,93	1,011
0.8	0.1	300	41	3,25	0,904	0,994
0.8	0.1	300	51	3,5	0,929	0,997
0.8	0.1	300	46	3,5	0,929	0,985
0.1	0.8	300	2	3,25	2,425	2,569
0.1	0.8	300	1	3	2,4	2,545
0.1	0.8	300	2	3,25	2,435	2,576
0.1	0.8	300	1	3	2,411	2,557
0.1	0.8	300	1	3	2,404	2,556
0.1	0.8	300	2	3,25	2,429	2,557
0.1	0.8	300	1	3	2,402	2,567
0.1	0.8	300	1	3	2,395	2,577
0.1	0.8	300	2	3,25	2,439	2,606
0.1	0.8	300	1	3	2,409	2,552
0.1	0.1	300	61	3,5	1,517	1,981
0.1	0.1	300	61	3,5	1,513	1,953
0.1	0.1	300	62	3,5	1,513	2,018
0.1	0.1	300	61	3,5	1,516	2,025
0.1	0.1	300	61	3,5	1,515	1,994
0.1	0.1	300	61	3,5	1,516	1,997
0.1	0.1	300	61	3,5	1,515	1,991
0.1	0.1	300	63	3,5	1,509	2,047
0.1	0.1	300	61	3,5	1,513	1,991
0.1	0.1	300	41	3,23	1,342	1,918

Tabla 4.6 Variante 1A caso β=0.5 (Fuente: elaboración propia).

Parámetro	Media	Desviación. E
$T_i = T_w$	46,50	6,851
T_p	3,4	0,129

Tabla 4.7 Variante 1B caso β=0.5 (Fuente: elaboración propia).

Parámetro	Media	Desviación. E
$T_i = T_w$	1,40	0,516
T_p	3,1	0,129

Tabla 4.8 Variante 1C caso β=0.5 (Fuente: elaboración propia).

Parámetro	Media	Desviación. E
$T_i = T_w$	59,30	6,464
T_p	3,473	0,085

Tabla 4.9 Variante 1, caso β=0.75 (Fuente: elaboración propia).

K_o	K_i	Nº NO mejora	$T_i = T_w$	T_p	J mod	J aprox
0.8	0.1	300	36	2	1,033	1,105
0.8	0.1	300	20	2,25	1,036	1,123
0.8	0.1	300	51	1,75	1,042	1,14
0.8	0.1	300	21	2,25	1,03	1,121
0.8	0.1	300	16	2,5	1,059	1,132
0.8	0.1	300	21	2,25	1,03	1,118
0.8	0.1	300	21	2,25	1,033	1,125
0.8	0.1	300	36	2	1,027	1,122
0.8	0.1	300	36	2	1,031	1,143
0.8	0.1	300	21	2,25	1,032	1,135
0.1	0.8	300	1	2,5	2,6	2,768
0.1	0.8	300	1	2,5	2,598	2,717
0.1	0.8	300	36	2	2,838	3,051
0.1	0.8	300	1	2,5	2,599	2,777
0.1	0.8	300	36	2	2,894	3,137
0.1	0.8	300	1	2,5	2,602	2,745
0.1	0.8	300	21	2,25	2,795	3,039
0.1	0.8	300	1	2,5	2,583	2,759
0.1	0.8	300	1	2,5	2,567	2,781
0.1	0.8	300	1	2,5	2,588	2,707
0.1	0.1	300	2	2,5	1,405	2,076
0.1	0.1	300	2	2,5	1,409	2,075
0.1	0.1	300	21	2,25	1,466	2,032
0.1	0.1	300	36	2	1,498	2,077
0.1	0.1	300	20	2,25	1,465	2,063
0.1	0.1	300	2	2,5	1,402	2,046
0.1	0.1	300	3	2,5	1,405	2,06
0.1	0.1	300	5	2,5	1,431	2,026
0.1	0.1	300	53	1,75	1,542	2,165
0.1	0.1	300	3	2,5	1,411	2,057

Tabla 4.10 Variante 1A caso β=0.75 (Fuente: elaboración propia).

Parámetro	Media	Desviación. E
$T_i = T_w$	27,9	11,18
T_p	2,15	0,21

Tabla 4.11 Variante 1B caso β=0.75 (Fuente: elaboración propia).

Parámetro	Media	Desviación. E
$T_i = T_w$	10	15,05
T_p	2,375	0,212

Tabla 4.12 Variante 1C caso β=0.75 (Fuente: elaboración propia).

Parámetro	Media	Desviación. E
$T_i = T_w$	14,70	17,764
T_p	2,325	0,2648

En esta primera variante, al proporcionar control al modelo únicamente sobre 2 parámetros, podemos observar que las reacciones a las variaciones de los pesos son considerablemente notables, especialmente en el caso de T_i y T_w, que para este escenario en concreto con un factor de suavización reducido ($\alpha = 0{,}1$) y un nivel de protección contra falta de existencias $\varepsilon = 1$, alcanza valores máximos y mínimos como podemos observar en la siguiente gráfica 4.1.

Como se puede apreciar, el modelo sugiere para este escenario en concreto, con una tasa de retorno media-baja, el empleo de controladores de inventario de valores muy reducidos en los casos en los que se dé mayor importancia a la variabilidad de los inventarios que a la de los pedidos y/o al stock neto medio, si la importancia residiera mayoritariamente en estos dos últimos aspectos o en uno de ellos, se sugiere todo lo contrario, aportar a estos controladores valores considerablemente elevados.

Por otra parte, si la tasa de retorno alcanza valores más elevados, los valores que nuestro modelo sugiere para estos controladores de inventario son totalmente diferentes a los casos anteriores con tasa de retorno media-baja. Cuando la importancia reside en la variabilidad de los inventarios deben tomar valores reducidos, pero sin llegar a rozar el mínimo como ocurría anteriormente. Asimismo, cuando tiene mayor peso la variabilidad de los pedidos y/o el stock neto medio, estos controladores de inventario se deben de reducir en gran medida con respecto al caso anterior en el que existía una tasa de retorno media-baja.

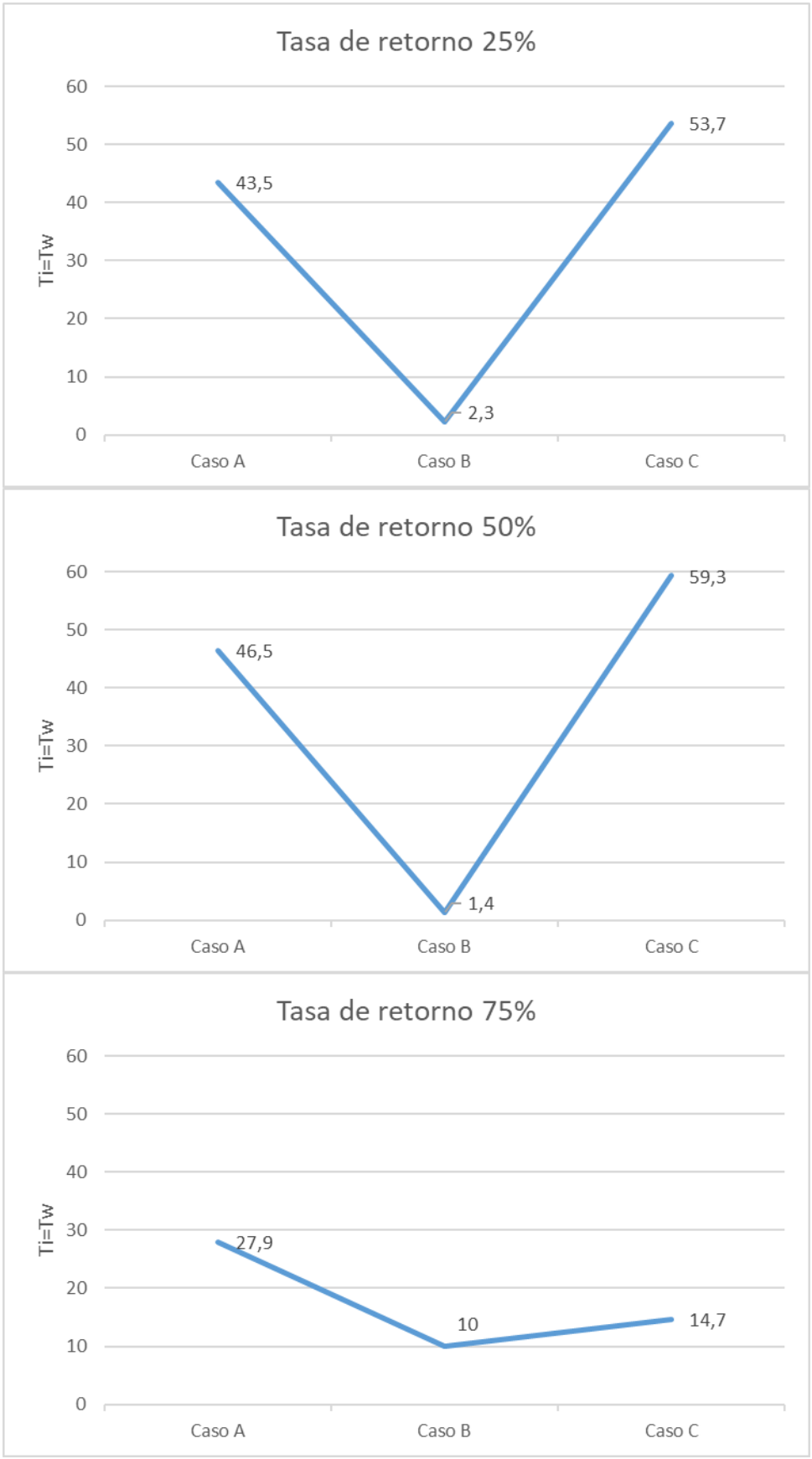

Figura 4.1 Valores de T_i y T_w en función de *Ko*, *Ki*, *Ks* y β para la variante 1 (Fuente: elaboración propia).

Con respecto a T_p, en la siguiente gráfica 4.2 representamos sus valores medios obtenidos en función de diferentes tasas de retorno y los comparamos con la aproximación utilizada en el artículo de referencia (Ecuación 12).

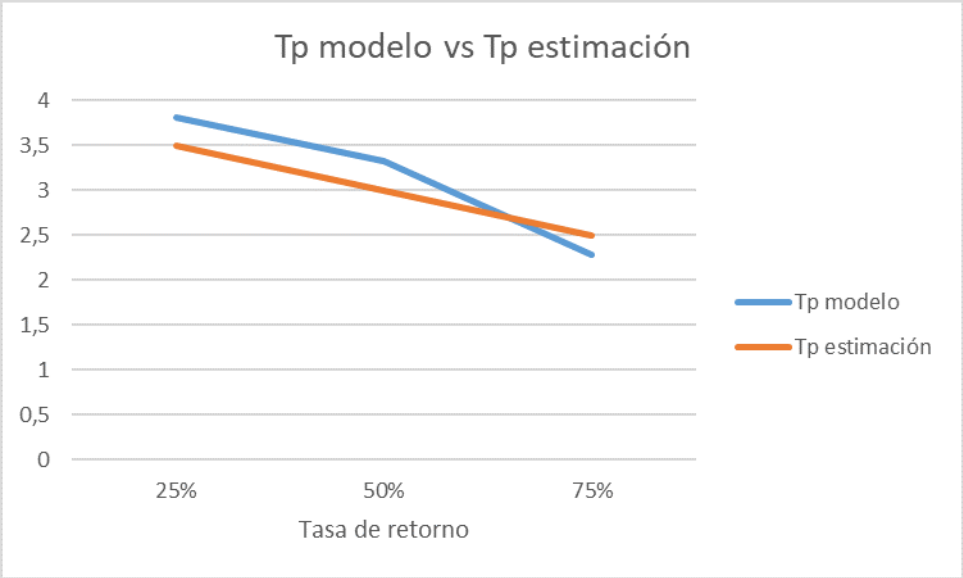

Figura 4.2 Comparación de valores de T_p para diferentes tasas de retorno en la variante 1 (Fuente: elaboración propia).

4.2 Resultados variante 2

Los resultados obtenidos con la segunda variante se recogen en las siguientes tablas, acompañadas por la media y desviación estándar de cada uno de los parámetros en cada escenario.

Tabla 4.13 Variante 2, caso β = 0.25 (Fuente: elaboración propia).

K_o	K_i	Nº NO mejora	$T_i = T_w$	T_p	α	ε	J mod	J aprox
0.8	0.1	90	41	3,75	0,1	1	0,746	0,827
0.8	0.1	90	36	4	0,1	0,7	0,746	0,812
0.8	0.1	90	41	4,5	0,1	0,2	0,74	0,813
0.8	0.1	90	41	2	0,1	2,7	0,738	0,821
0.8	0.1	90	36	4,5	0,1	0,2	0,742	0,814
0.8	0.1	90	46	2,25	0,1	2,5	0,743	0,822
0.8	0.1	90	46	4	0,1	0,8	0,75	0,826
0.8	0.1	90	37	2,5	0,1	2,2	0,748	0,823
0.8	0.1	90	46	2,5	0,1	2,3	0,748	0,834
0.8	0.1	90	41	4	0,1	0,7	0,736	0,83
0.1	0.8	90	61	4	0,8	0,5	2,09	2,22
0.1	0.8	90	51	4,5	0,7	0	2,094	2,231
0.1	0.8	90	61	2	0,7	2,5	2,088	2,201
0.1	0.8	90	61	3	0,7	1,5	2,098	2,231
0.1	0.8	90	51	2,75	0,7	1,8	2,101	2,204
0.1	0.8	90	46	4,5	0,6	0	2,097	2,241
0.1	0.8	90	56	3	0,7	1,5	2,095	2,213
0.1	0.8	90	51	4,5	0,6	0	2,096	2,199
0.1	0.8	90	51	2,5	0,8	2	2,089	2,225
0.1	0.8	90	51	4,25	0,8	0,2	2,086	2,241
0.1	0.1	90	53	4,5	0,6	0	1,1	1,651
0.1	0.1	90	61	3,5	0,6	1	1,093	1,632
0.1	0.1	90	61	3,5	0,6	1	1,093	1,628
0.1	0.1	90	61	2,5	0,6	2	1,099	1,643
0.1	0.1	90	56	3,25	0,8	1,2	1,09	1,709
0.1	0.1	90	61	3,5	0,5	1	1,092	1,62
0.1	0.1	90	56	1,75	0,8	2,7	1,09	1,699
0.1	0.1	90	59	2,25	0,8	2,2	1,091	1,704
0.1	0.1	90	61	4,5	0,5	0	1,091	1,628
0.1	0.1	90	61	4,5	0,5	0	1,088	1,591

Tabla 4.14 Variante 2A caso β =0.25 (Fuente: elaboración propia).

Parámetro	Media	Desv. Estándar
$T_i = T_w$	41,1	3,956
T_p	3,4	0,973
α	0,1	0
ε	1,33	0,982

Tabla 4.15 Variante 2B caso β = 0.25 (Fuente: elaboración propia).

Parámetro	Media	Desv. Estándar
$T_i = T_w$	54	5,374
T_p	3,5	0,950
α	0,71	0,073
ε	1	0,959

Tabla 4.16 Variante 2C caso β = 0.25 (Fuente: elaboración propia)..

Parámetro	Media	Desv. Estándar
$T_i = T_w$	59	2,943
T_p	3,375	0,973
α	0,63	0,125
ε	1,11	0,957

Tabla 4.17 Variante 2 caso β = 0.5 (Fuente: elaboración propia).

K_o	K_i	Nº NO mejora	$T_i = T_w$	T_p	α	ε	J mod	J aprox
0.8	0.1	90	34	3,75	0,1	0,5	0,915	0,982
0.8	0.1	90	56	2	0,2	2,2	0,924	1,009
0.8	0.1	90	41	3,25	0,1	1	0,91	0,999
0.8	0.1	90	43	4	0,1	0,3	0,912	0,996
0.8	0.1	90	47	2,25	0,1	2	0,907	0,997
0.8	0.1	90	36	3,25	0,1	1	0,909	0,994
0.8	0.1	90	43	2,5	0,1	1,8	0,908	0,987
0.8	0.1	90	33	3,25	0,1	1	0,911	1,004
0.8	0.1	90	41	4,25	0,1	0	0,909	0,982
0.8	0.1	90	46	2	0,1	2,3	0,911	1,001
0.1	0.8	90	43	3,5	0,7	0,3	2,253	2,393
0.1	0.8	90	56	1,75	0,8	1,9	2,265	2,416
0.1	0.8	90	41	3	0,6	1	2,259	2,399
0.1	0.8	90	31	3,5	0,5	0,5	2,257	2,464
0.1	0.8	90	61	2,25	0,6	1,7	2,251	2,373
0.1	0.8	90	36	1,25	0,8	2,5	2,265	2,408
0.1	0.8	90	49	2,5	0,6	1,4	2,262	2,374
0.1	0.8	90	36	1,5	0,5	2,5	2,269	2,489
0.1	0.8	90	56	2,25	0,6	1,6	2,244	2,415
0.1	0.8	90	56	1,75	0,5	2,2	2,252	2,404
0.1	0.1	90	51	2,25	0,6	1,6	1,16	1,765
0.1	0.1	90	54	1,75	0,5	2,2	1,178	1,76
0.1	0.1	90	36	2,5	0,6	1,4	1,185	1,797
0.1	0.1	90	56	3,25	0,7	0,5	1,168	1,789
0.1	0.1	90	56	2,5	0,5	1,4	1,169	1,77
0.1	0.1	90	35	3,5	0,6	0,4	1,181	1,811
0.1	0.1	90	56	3,25	0,5	0,7	1,183	1,752
0.1	0.1	90	36	1	0,7	2,8	1,177	1,823
0.1	0.1	90	51	3,75	0,6	0,1	1,164	1,768
0.1	0.1	90	51	2,75	0,8	0,9	1,186	1,801

Tabla 4.18 Variante 2A caso β = 0.5 (Fuente: elaboración propia).

Parámetro	Media	Desv. Estándar
$T_i = T_w$	42	6,847
T_p	3,05	0,823
α	0,11	0,031
ε	1,21	0,821

Tabla 4.19 Variante 2B caso β = 0.5 (Fuente: elaboración propia).

Parámetro	Media	Desv. Estándar
$T_i = T_w$	46,5	10,490
T_p	2,325	0,799
α	0,62	0,113
ε	1,56	0,772

Tabla 4.20 Variante 2C caso β = 0.5 (Fuente: elaboración propia).

Parámetro	Media	Desv. Estándar
$T_i = T_w$	48,2	8,891
T_p	2,65	0,843
α	0,61	0,099
ε	1,2	0,848

Tabla 4.21 Variante 2 caso $\beta = 0.75$ (Fuente: elaboración propia).

K_o	K_i	Nº NO mejora	$T_i = T_w$	T_p	α	ε	J mod	J aprox
0.8	0.1	90	51	1	0,2	1,4	1,025	1,1
0.8	0.1	90	31	1,5	0,2	1,3	1,027	1,125
0.8	0.1	90	31	0,5	0,2	2,3	1,029	1,112
0.8	0.1	90	31	1,5	0,3	1,099	1,031	1,111
0.8	0.1	90	31	0,75	0,2	2,1	1,027	1,113
0.8	0.1	90	26	2,75	0,1	0,4	1,033	1,124
0.8	0.1	90	31	0,75	0,1	2,3	1,028	1,115
0.8	0.1	90	26	1,5	0,293	1,3	1,032	1,124
0.8	0.1	90	30	1,75	0,2	1,09	1,028	1,121
0.8	0.1	90	41	2	0,1	0,899	1,03	1,118
0.1	0.8	90	11	1	0,6	1,8	2,545	2,752
0.1	0.8	90	11	1,25	0,5	1,7	2,579	2,705
0.1	0.8	90	7	2,25	0,7	0,6	2,549	2,76
0.1	0.8	90	11	1,25	0,6	1,6	2,554	2,739
0.1	0.8	90	6	0,5	0,7	2,4	2,552	2,736
0.1	0.8	90	6	2,25	0,8	0,5	2,555	2,771
0.1	0.8	90	16	0,75	0,5	2,1	2,557	2,725
0.1	0.8	90	11	0,5	0,7	2,2	2,567	2,756
0.1	0.8	90	16	1,75	0,6	0,9	2,573	2,78
0.1	0.8	90	11	2,25	0,5	0,7	2,537	2,739
0.1	0.1	90	11	1,5	0,6	1,3	1,314	1,959
0.1	0.1	90	11	1	0,481	1,9	1,309	1,992
0.1	0.1	90	3	2	0,5	1,09	1,337	2,078
0.1	0.1	90	1	2,75	0,1	0,6	1,341	2,113
0.1	0.1	90	1	1,25	0,1	2,1	1,335	2,107
0.1	0.1	90	1	2,75	0,1	0,6	1,332	2,123
0.1	0.1	90	6	0,75	0,8	2	1,316	2,061
0.1	0.1	90	16	2,25	0,581	0,4	1,324	1,952
0.1	0.1	90	11	0,5	0,7	2,2	1,33	1,992
0.1	0.1	90	1	2,25	0,1	1,09	1,335	2,1

Tabla 4.22 Variante 2A caso $\beta = 0.75$ (Fuente: elaboración propia).

Parámetro	Media	Desv. Estándar
$T_i = T_w$	32,9	7,563
T_p	1,4	0,679
α	0,1893	0,072
ε	1,4188	0,629

Tabla 4.23 Variante 2B caso $\beta = 0.75$ (Fuente: elaboración propia).

Parámetro	Media	Desv. Estándar
$T_i = T_w$	10,6	3,565
T_p	1,375	0,709
α	0,62	0,103
ε	1,45	0,713

Tabla 4.24 Variante 2C caso β = 0.75 (Fuente: elaboración propia).

Parámetro	Media	Desv. Estándar
$T_i = T_w$	6,2	5,613
T_p	1,7	0,814
α	0,4062	0,278
ε	1,328	0,680

El primer comportamiento a destacar en esta segunda variante es que a diferencia de en la primera, los valores de $T_i = T_w$ no se ven altamente afectados por el aumento del peso de la variabilidad de los inventarios, sin embargo, sí que se aprecian estas alteraciones en los valores de α como podemos observar en las siguientes gráficas:

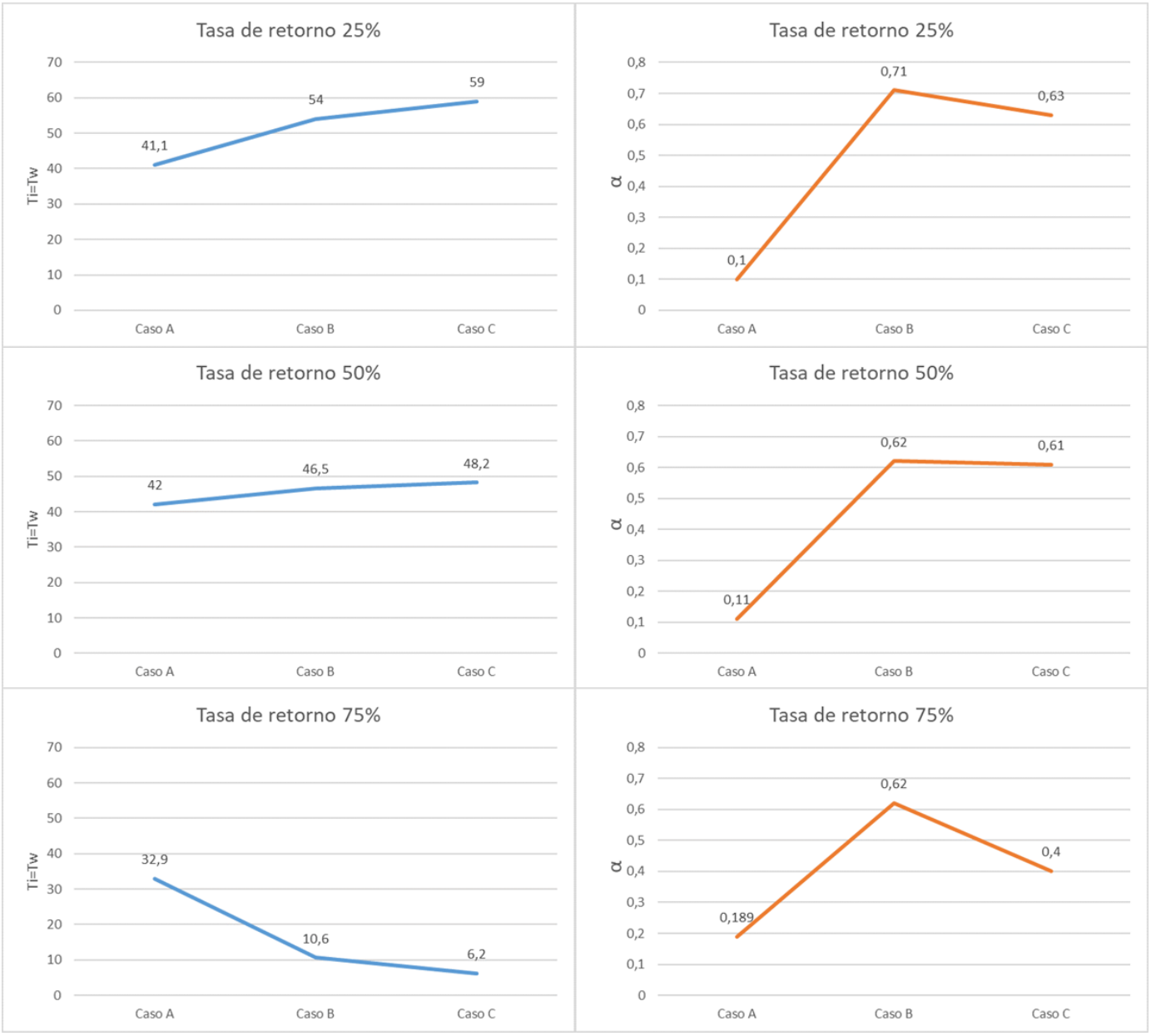

Figura 4.3 Valores de $T_i = T_w$ y α en función de Ko, Ki, Ks y β para la variante 2 (Fuente: elaboración propia).

En cuanto a T_p, su aproximación a la estimación del modelo de referencia es mayor cuanto menor es la tasa de retorno como podemos ver representado en la figura 4.4:

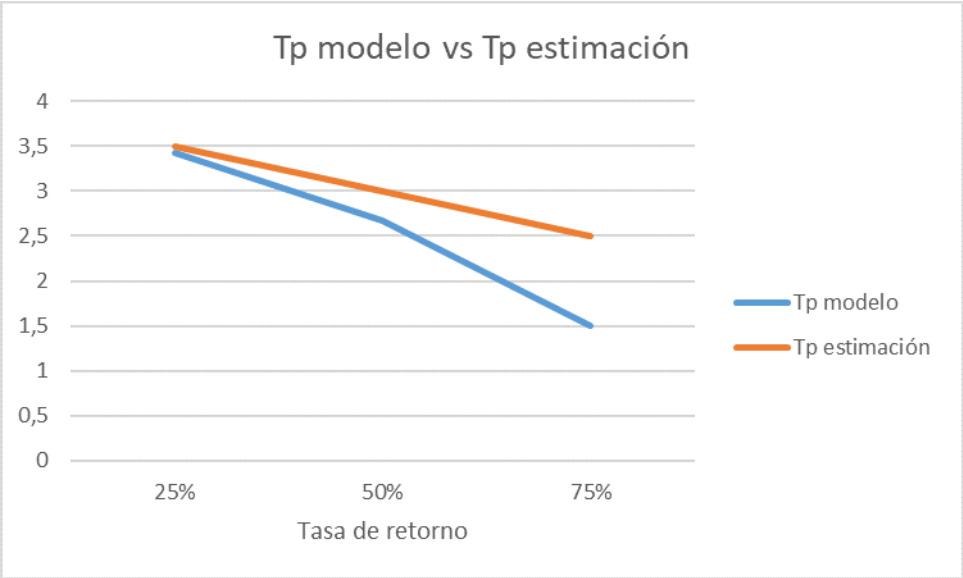

Figura 4.4 Comparación de valores de T_p para diferentes tasas de retorno en la variante 2 (Fuente: elaboración propia).

Un último comportamiento muy interesante a destacar es la relación de proporcionalidad inversa que se observa entre T_p y ε al dejar total control al modelo sobre estos parámetros. Podemos ver este comportamiento reflejado en la siguiente gráfica 4.5:

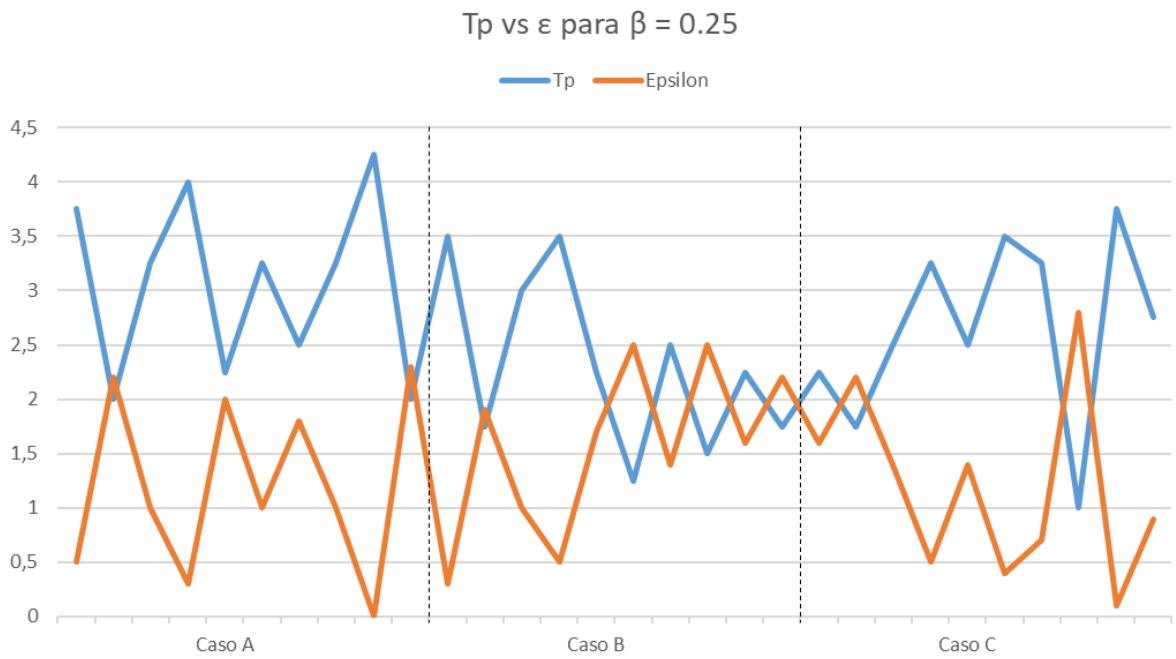

Figura 4.5 Comparación de valores de T_p y ε en la variante 2 (Fuente: elaboración propia).

5 Conclusiones

La configuración de las cadenas de suministro de ciclo cerrado o híbrido es un gran reto para las industrias actuales, uno de los pilares fundamentales en la optimización de estas configuraciones es la correcta definición de políticas de pedido mediante las cuales se controla en gran medida la producción de estas cadenas de suministro.

Cada vez más se está innovando en este campo y surgen sistemas como las políticas de pedidos proporcionales (POUT), que nos permiten regular la producción minimizando los efectos adversos como el efecto látigo. Sin embargo, los estudios que encontramos en la literatura sobre estas políticas para este tipo de cadenas de suministro aún son considerablemente escasos, lo que se traduce en la existencia de un amplio abanico de mejora.

Concretamente en el artículo Cannella et al. (2021) se estudia el funcionamiento de estas políticas proporcionales en un sistema híbrido de fabricación y refabricación, mostrando su mejor rendimiento frente a unas políticas de pedido tradicionales y sugiriendo diferentes configuraciones de sus parámetros en función de distintos escenarios.

Con este proyecto se ha tratado de diseñar un modelo de optimización basado en una búsqueda heurística que nos aporte soluciones concretas para escenarios definidos. El algoritmo implementado ha mejorado el rendimiento dinámico de la cadena de suministro, obteniendo valores ligeramente menores de la función de coste con respecto a los publicados en Cannella et al. (2021). Además, se han identificado algunas tendencias claras a la hora de configurar los parámetros de control de la cadena dependiendo de la estructura de dicha función de coste, estando algunas de estas tendencias alienadas con literatura, mientras que otras son totalmente novedosas.

Aunque, en cierta medida se ha logrado el objetivo, encontramos algunas debilidades en el modelo que con su estudio en futuros proyectos pueden dar lugar a posibles mejoras o nuevas líneas de investigación.

El primer posible aspecto de continuación de este proyecto, sería el análisis del método heurístico empleado, en este trabajo se ha recurrido a una adaptación del *Iterated Greedy* debido a su versatilidad, lo que se ha traducido en una completa adaptación a nuestras necesidades. Sin embargo, el análisis de diferentes opciones de heurísticas no ha formado parte de los objetivos de este trabajo, en consecuencia, es muy probable la existencia de otros algoritmos que mejoren el rendimiento de la búsqueda de resultados.

Por otro lado encontramos que, aunque al aumentar el número de parámetros sobre los que el modelo tiene control disminuye la convergencia de los resultados, se siguen observando ciertas relaciones inesperadas entre variables. Al tratarse del estudio de un sistema dinámico, no podemos comprender la razón última de estos comportamientos de forma directa, lo cual, si se podría conseguir mediante un análisis más detallado, alcanzando así a comprender mejor este sistema híbrido con modelos de reposición POUT a fin de poder perfeccionar el modelo presentado en este trabajo o crear algún otro.

En definitiva, en este proyecto se ha diseñado un modelo para optimizar la dinámica de una cadena de suministro circular mediante controladores de inventario y otros parámetros de control, y se muestran dos posibles variantes del modelo a utilizar en función de las variables sobre las que tengamos control. En general se da a conocer que las políticas de POUT funcionan más allá de las cadenas de suministro tradicionales y que se pueden configurar de manera que optimicen su rendimiento para cada cadena de suministro y escenario en concreto.

Bibliografía

[Bouchery et al., 2017] Bouchery, Y., Corbett, C. J., Fransoo, J. C., and Tan, T. (2017). *Sustainable Supply Chains*, Springer series in supply chain management 4, doi 10.1007/978-3-319-29791-0_17.

[Cannella et al., 2016] Cannella, S., Bruccoleri, M., and Framinan, J. (2016). Closed-loop supply chains: What reverse logistics factors influence performance? *International Journal of Production Economics*, 175:35–46.

[Cannella et al., 2021] Cannella, S., Ponte, B., Dominguez, R., and Framinan, J. M. (2021). Proportional order-up-to policies for closed-loop supply chains: the dynamic effects of inventory controllers. *International Journal of Production Research*, 59:11:3323–3337.

[Dominguez et al., 2021] Dominguez, R., Cannella, S., and Framinan, J. M. (2021). *Remanufacturing configuration in complex supply chains*. *Omega*, 101.

[Framinan et al., 2014] Framinan, J. M., Leisten, R., and García, R. R. (2014). *Manufacturing scheduling systems: An integrated view on models, methods and tools.* Springer-Verlag London Ltd. https://doi.org/10.1007/978-1-4471-6272-8.

[García and Aníbal, 2021] García, M. and Aníbal, L. (2021). *Cadena de suministro.* Barcelona : Marge Books.

[Genovese et al., 2017] Genovese, A., Acquaye, A. A., Figueroa, A., and Koh, S. L. (2017). Sustainable supply chain management and the transition towards a circular economy: Evidence and some applications. *Omega*, 66B:344–357.

[Govindan et al., 2015] Govindan, K., Soleimani, H., and Kannan, D. (2015). Reverse logistics and closed-loop supply chain: A comprehensive review to explore the future. *European Journal of Operational Research*, 240:603–626.

[Mecalux, 2022] Mecalux (15 de Noviembre de 2022). https://www.mecalux.es/blog/efecto-latigo-bullwhip-effect.

[Mishra et al., 2022] Mishra, A., Dutta, P., Jayasankar, S., Jain, P., and Mathiyazhagan, K. (2022). A review of reverse logistics and closed-loop supply chains in the perspective of circular

economy, *Benchmarking: An International Journal*, Vol. ahead-of-print No. ahead-of-print. https://doi.org/10.1108/bij-11-2021-0669.

[Pérez, 2020] Pérez, A. P. (2020). Trabajo de Fin de Grado. *Impacto de la capacidad sobre el efecto Bullwhip en cadenas de suministro.*

[Roldán, 2017] Roldán, P. N. (2017). *Cadena de suministro*. Economipedia.com, Url: https://economipedia.com/definiciones/cadena-de-suministro.html.

ANEXO I

1 Códigos

Código 5.1 Implementación en Python modelo matemático Cannella et al. (2021). Función evaluarJ (Fuente: Elaboración propia).

```
print("Inicio de simulación")
import numpy as np
from numpy import var
from numpy import mean

#DECLARAMOS CONSTANTES
E = 11 #Número de ejecuciones de la simulación
T = 2100 #Número de periodos de cada simulación
tc = 16
tm = 4
tr = 2
alpha = 0.4
epsilon = 1
ko=0.5
ki=1-ko
#DECLARAMOS VARIABLES DE LAS DISTRIBUCIONES NORMALES
mu = 100
sigma = 30
beta = 0.5
xi = 0.05

#PARÁMETROS DE DECISIÓN
Ti = 1
Tw = 1
#-----------------------------------------------------------------

#VARIABLES VECTORIALES, CAMBIAN DE VALOR EN CADA ITERACIÓN
d = []  #Demanda real
```

```
r = []  #Devoluciones
pd = [] #Previsión demanda
ss = [] #Stock de seguridad
tw = [] #Trabajo en curso objetivo
mc = [] #Unidades nuevas que finalizan en t
rc = [] #Unidades refabricadas que finalizan en t
si = [] #Stock inicial en t
ns = [] #Stock neto en t
o = []  #Orden de producción en t
w = []  #Trabajo en curso en t
j = []  #resultado de cada ejecución


#INICIO DE SIMULACIÓN
tp = 3
#tp = (1-beta)*tm + beta*tr #Estimación plazo de entrega
e = 1
while e<E:
    t = 0
    d.clear()
    r.clear()
    pd.clear()
    ss.clear()
    tw.clear()
    mc.clear()
    rc.clear()
    si.clear()
    ns.clear()
    o.clear()
    w.clear()
    while t<T:
        #DEMANDA DEL CLIENTE t
        xt = np.random.normal(loc = mu, scale = sigma, size = None)
        if xt > 0:
            d.append(xt)
        else:
            d.append(0)
        #print("La demanda en",t,"es",d[t])
        #DEVOLUCIONES DEL CLIENTE EN t
        zt = np.random.normal(loc = beta, scale = xi, size = None)
        if zt > 0 and zt < 1:
            yt = zt
        elif zt < 0:
            yt = 0
        else:
            yt = 1
        if t>=tc:
            r.append(yt*d[t-tc])
        else:
            r.append(0)
```

```
        #print("Las devoluciones en",t,"es",r[t])

        #PREVISIÓN DE LA DEMANDA EN t
        if t>0:
            pd.append((alpha*d[t])+(1-alpha)*pd[t-1])
        else:
            pd.append(alpha*d[t])
        #print ("La prevision de la demanda en",t," es",pd[t])

        #STOCK DE SEGURIDAD EN t
        ss.append(epsilon*pd[t])
        #print("El stock de seguridad en",t,"es",ss[t])

        #TRABAJO EN CURSO OBJETIVO EN t
        tw.append(tp*pd[t])
        #print("El trabajo en curso objetivo en",t,"es",tw[t])

        #UNIDADES NUEVAS QUE FINALIZAN EN EL PERIODO t
        if t-tm-1 < 0:
            mc.append(0)
        else:
            mc.append(o[t-tm-1])
        #print("Las unidades nuevas que finalizan en",t,"es",mc[t])

        #UNIDADES REFABRICADAS QUE FINALIZAN EN EL PERIODO t
        if t-tr-1 < 0:
            rc.append(0)
        else:
            rc.append(r[t-tr-1])
        #print("Las unidades refab que finalizan en",t,"es",rc[t])

        #STOCK INICIAL EN t
        if t<1:
            si.append(0)
        else:
            si.append(ns[t-1]+mc[t]+rc[t])
        #print("El stock inicial en",t,"es",si[t])

        #STOCK NETO EN t
        ns.append(si[t]-d[t])
        #print("El stock neto en",t,"es",ns[t])

        #TRABAJO EN CURSO EN t
        if t<1:
            w.append(0)
        else:
            w.append(w[t-1]+(o[t-1]-mc[t])+(r[t-1]-rc[t]))
        #print("El trabajo en curso en",t,"es",w[t])

        #ORDEN DE PRODUCCIÓN EN t
```

```
        aux = ((pd[t]-rc[t])+(1/Ti)*(ss[t]-ns[t])+(1/Tw)*(tw[t]-w[t]))
        if aux > 0:
            o.append(aux)
        else:
            o.append(0)
        #print("Las ordenes de prod en",t,"es",o[t])

        t = t+1
    #CÁLCULO PARÁMETROS DE RENDIMIENTO
    varo = var(o[99:])
    vard = var(d[99:])
    varns = var(ns[99:])
    OVR = varo/vard
    IVR = varns/vard
    j.append((ko*np.sqrt(OVR))+(ki*np.sqrt(IVR)))
    #print("OVR:",OVR,"IVR:",IVR,"VAR(O):",varo,"VAR(NS):",varns,"VAR(D)
        :",vard)
    print("El valor de J de la ejecución ",e,"es: ",j[e-1])
    e = e+1
J = mean(j)
print(J)
```

Código 5.2 Adaptación modelo matemático. Función evaluarJmod (Fuente: Elaboración propia).

```
print("Inicio de simulación")
import numpy as np
from numpy import var
from numpy import mean

#DECLARAMOS CONSTANTES
E = 11 #Número de ejecuciones de la simulación
T = 2100 #Número de periodos de cada simulación
tc = 16
tm = 4
tr = 2
alpha = 0.1
epsilon = 1
ko=0.33
ki=0.33
ks=1-ko-ki
#DECLARAMOS VARIABLES DE LAS DISTRIBUCIONES NORMALES
mu = 100
sigma = 30
beta = 0.5
xi = 0.05

#PARÁMETROS DE DECISIÓN
Ti = 4
Tw = 3
```

```
dns = 10 # Porcentaje de demanda no satisfecha que podemos asumir
#-------------------------------------------------------

#VARIABLES VECTORIALES, CAMBIAN DE VALOR EN CADA ITERACIÓN
d = []  #Demanda real
r = []  #Devoluciones
pd = [] #Previsión demanda
ss = [] #Stock de seguridad
tw = [] #Trabajo en curso objetivo
mc = [] #Unidades nuevas que finalizan en t
rc = [] #Unidades refabricadas que finalizan en t
si = [] #Stock inicial en t
ns = [] #Stock neto en t
o = []  #Orden de producción en t
w = []  #Trabajo en curso en t
jmod = []  #resultado de cada ejecución


#INICIO DE SIMULACIÓN
tp = 3.25  #Estimación plazo de entrega
e = 1
while e<E:
    t = 0
    d.clear()
    r.clear()
    pd.clear()
    ss.clear()
    tw.clear()
    mc.clear()
    rc.clear()
    si.clear()
    ns.clear()
    o.clear()
    w.clear()
    while t<T:
        #DEMANDA DEL CLIENTE t
        xt = np.random.normal(loc = mu, scale = sigma, size = None)
        if xt > 0:
            d.append(xt)
        else:
            d.append(0)
        #print("La demanda en",t,"es",d[t])
        #DEVOLUCIONES DEL CLIENTE EN t
        zt = np.random.normal(loc = beta, scale = xi, size = None)
        if zt > 0 and zt < 1:
            yt = zt
        elif zt < 0:
            yt = 0
        else:
            yt = 1
```

```
        if t>=tc:
            r.append(yt*d[t-tc])
        else:
            r.append(0)
        #print("Las devoluciones en",t,"es",r[t])

        #PREVISIÓN DE LA DEMANDA EN t
        if t>0:
            pd.append((alpha*d[t])+(1-alpha)*pd[t-1])
        else:
            pd.append(alpha*d[t])
        #print ("La prevision de la demanda en",t," es",pd[t])

        #STOCK DE SEGURIDAD EN t
        ss.append(epsilon*pd[t])
        #print("El stock de seguridad en",t,"es",ss[t])

        #TRABAJO EN CURSO OBJETIVO EN t
        tw.append(tp*pd[t])
        #print("El trabajo en curso objetivo en",t,"es",tw[t])

        #UNIDADES NUEVAS QUE FINALIZAN EN EL PERIODO t
        if t-tm-1 < 0:
            mc.append(0)
        else:
            mc.append(o[t-tm-1])
        #print("Las unidades nuevas que finalizan en",t,"es",mc[t])

        #UNIDADES REFABRICADAS QUE FINALIZAN EN EL PERIODO t
        if t-tr-1 < 0:
            rc.append(0)
        else:
            rc.append(r[t-tr-1])
        #print("Las unidades refab que finalizan en",t,"es",rc[t])

        #STOCK INICIAL EN t
        if t<1:
            si.append(0)
        else:
            si.append(ns[t-1]+mc[t]+rc[t])
        #print("El stock inicial en",t,"es",si[t])

        #STOCK NETO EN t
        ns.append(si[t]-d[t])
        #print("El stock neto en",t,"es",ns[t])

        #TRABAJO EN CURSO EN t
        if t<1:
            w.append(0)
        else:
```

```
            w.append(w[t-1]+(o[t-1]-mc[t])+(r[t-1]-rc[t]))
        #print("El trabajo en curso en",t,"es",w[t])

        #ORDEN DE PRODUCCIÓN EN t
        aux = ((pd[t]-rc[t])+(1/Ti)*(ss[t]-ns[t])+(1/Tw)*(tw[t]-w[t]))
        if aux > 0:
            o.append(aux)
        else:
            o.append(0)
        #print("Las ordenes de prod en",t,"es",o[t])

        t = t+1

    #VALIDACIÓN DE LA SOLUCIÓN
    neg = 0
    for num in ns[99:]:
        if num < 0:
            neg += 1
    porcneg=(neg/2000)*100
    if porcneg < dns:
        #CÁLCULO PARÁMETROS DE RENDIMIENTO
        varo = var(o[99:])
        vard = var(d[99:])
        varns = var(ns[99:])
        OVR = varo/vard
        IVR = varns/vard
        nsm = mean(ns[99:])
        jmod.append((ko*np.sqrt(OVR))+(ki*np.sqrt(IVR))+(ks*(nsm/100)))
        #print("OVR:",OVR,"IVR:",IVR,"VAR(O):",varo,"VAR(NS):",varns,"
            VAR(D):",vard)
        print("El valor de Jmod de la ejecución ",e,"es: ",jmod[-1])
    e = e+1
if len(jmod)<9:
    print("solucion no factible")
else:
    Jmod = mean(jmod)
    print(Jmod)
```

Código 5.3 Variante 1 (Fuente: Elaboración propia).

```
def evaluarJmod(Ti,Tw,tp):
    import numpy as np
    from numpy import var
    from numpy import mean
    #DECLARAMOS CONSTANTES
    E = 11 #Número de ejecuciones de la simulación
    T = 2100 #Número de periodos de cada simulación
    tc = 16
    tm = 4
```

```
tr = 2
alpha = 0.1
epsilon = 1
ko=0.7
ki=0.15
ks=1-ko-ki
dns = 10 # Porcentaje de demanda no satisfecha que podemos asumir
#DECLARAMOS VARIABLES DE LAS DISTRIBUCIONES NORMALES
mu = 100
sigma = 30
beta = 0.5
xi = 0.05
#------------------------------
#VARIABLES VECTORIALES, CAMBIAN DE VALOR EN CADA ITERACIÓN
d = []  #Demanda real
r = []  #Devoluciones
pd = [] #Previsión demanda
ss = [] #Stock de seguridad
tw = [] #Trabajo en curso objetivo
mc = [] #Unidades nuevas que finalizan en t
rc = [] #Unidades refabricadas que finalizan en t
si = [] #Stock inicial en t
ns = [] #Stock neto en t
o = []  #Orden de producción en t
w = []  #Trabajo en curso en t
j = []  #resultado de cada ejecución
#INICIO DE SIMULACIÓN
e = 1
while e<E:
    t = 0
    d.clear()
    r.clear()
    pd.clear()
    ss.clear()
    tw.clear()
    mc.clear()
    rc.clear()
    si.clear()
    ns.clear()
    o.clear()
    w.clear()
    while t<T:
        #DEMANDA DEL CLIENTE t
        xt = np.random.normal(loc = mu, scale = sigma, size = None)
        if xt > 0:
            d.append(xt)
        else:
            d.append(0)
        #DEVOLUCIONES DEL CLIENTE EN t
        zt = np.random.normal(loc = beta, scale = xi, size = None)
```

```
            if zt > 0 and zt < 1:
                yt = zt
            elif zt < 0:
                yt = 0
            else:
                yt = 1
            if t>=tc:
                r.append(yt*d[t-tc])
            else:
                r.append(0)
            #PREVISIÓN DE LA DEMANDA EN t
            if t>0:
                pd.append((alpha*d[t])+(1-alpha)*pd[t-1])
            else:
                pd.append(alpha*d[t])
            #STOCK DE SEGURIDAD EN t
            ss.append(epsilon*pd[t])
            #TRABAJO EN CURSO OBJETIVO EN t
            tw.append(tp*pd[t])
            #UNIDADES NUEVAS QUE FINALIZAN EN EL PERIODO t
            if t-tm-1 < 0:
                mc.append(0)
            else:
                mc.append(o[t-tm-1])
            #UNIDADES REFABRICADAS QUE FINALIZAN EN EL PERIODO t
            if t-tr-1 < 0:
                rc.append(0)
            else:
                rc.append(r[t-tr-1])
            #STOCK INICIAL EN t
            if t<1:
                si.append(0)
            else:
                si.append(ns[t-1]+mc[t]+rc[t])
            #STOCK NETO EN t
            ns.append(si[t]-d[t])
            #TRABAJO EN CURSO EN t
            if t<1:
                w.append(0)
            else:
                w.append(w[t-1]+(o[t-1]-mc[t])+(r[t-1]-rc[t]))
            #ORDEN DE PRODUCCIÓN EN t
            aux = ((pd[t]-rc[t])+(1/Ti)*(ss[t]-ns[t])+(1/Tw)*(tw[t]-w[t])
                )
            if aux > 0:
                o.append(aux)
            else:
                o.append(0)
            t = t+1
```

```
        #VALIDACIÓN DE LA SOLUCIÓN
        neg = 0
        for num in ns[99:]:
            if num < 0:
                neg += 1
        porcneg=(neg/2000)*100
        if porcneg < dns:
            #CÁLCULO PARÁMETROS DE RENDIMIENTO
            varo = var(o[99:])
            vard = var(d[99:])
            varns = var(ns[99:])
            OVR = varo/vard
            IVR = varns/vard
            nsm = mean(ns[99:])
            j.append((ko*np.sqrt(OVR))+(ki*np.sqrt(IVR))+(ks*(nsm/100)))
        e = e+1
    if len(j)<9:
        J = 10000000
    else:
        J = mean(j)
    return J
#-----------------------------------------------
import random
    Jmin =10000000
    while Jmin==10000000: # para evitar escenario incompatible
        Ti=random.randint(1,64)
        Tw=Ti
        Tp=random.uniform(0.5,6)
        Jmin=evaluarJ1(Ti,Tw,Tp)
    eliminar=5
    nomejora=0
    while nomejora < 300:
        mejora=0
        eliminarant=eliminar
        while eliminarant==eliminar:
            eliminar=random.randint(0,1)
        if eliminar== 0:
            Tiaux=1
            Twaux=1
            while Tiaux < 62:
                J=evaluarJ1(Tiaux,Twaux,Tp)
                if J<Jmin:
                    print("bucle 1",J)
                    Jmin = J
                    Ti=Tiaux
                    Tw=Twaux
                    mejora=1
                Tiaux = Tiaux + 5
                Twaux = Twaux + 5
            if mejora == 1:
```

```
            if Ti-10>0 and Ti+10<64:
                Tiaux=Ti-10
                Twaux=Tw-10
                Cotasup=Ti+10
                while Tiaux<Cotasup:
                     J=evaluarJ1(Tiaux,Twaux,Tp)
                     if J<Jmin:
                         print("bucle 1 a",J)
                         Jmin = J
                         Ti=Tiaux
                         Tw=Twaux
                     Tiaux = Tiaux + 1
                     Twaux = Twaux + 1
            elif Ti-10<0:
                Tiaux=1
                Twaux=1
                Cotasup=Ti+10
                while Tiaux<Cotasup:
                     J=evaluarJ1(Tiaux,Twaux,Tp)
                     if J<Jmin:
                         print("bucle 1 b",J)
                         Jmin = J
                         Ti=Tiaux
                         Tw=Twaux
                     Tiaux = Tiaux + 1
                     Twaux = Twaux + 1
            elif Ti+10>64:
                Tiaux=Ti-10
                Twaux=Tw-10
                Cotasup=64
                while Tiaux<Cotasup:
                     J=evaluarJ1(Tiaux,Twaux,Tp)
                     if J<Jmin:
                         print("bucle 1 c",J)
                         Jmin = J
                         Ti=Tiaux
                         Tw=Twaux
                     Tiaux = Tiaux + 1
                     Twaux = Twaux + 1


        elif eliminar== 1:
            Tpaux=0.5
            while Tpaux < 6:
                J=evaluarJ1(Ti,Tw,Tpaux)
                if J<Jmin:
                    print("bucle 3",J)
                    Jmin = J
                    Tp=Tpaux
                    mejora=1
```

```
                Tpaux = Tpaux + 0.25
        if mejora == 0:
            nomejora = nomejora + 1
        elif mejora == 1:
           nomejora = 0
    print("Ti=",Ti,"Tw=",Tw,"Tp=",Tp)
```

Código 5.4 Variante 2 (Fuente: Elaboración propia).

```
def evaluarJ2(Ti,Tw,tp,alpha,epsilon):
    import numpy as np
    from numpy import var
    from numpy import mean
    #DECLARAMOS CONSTANTES
    E = 11 #Número de ejecuciones de la simulación
    T = 2100 #Número de periodos de cada simulación
    tc = 16
    tm = 4
    tr = 2
    ko=0.33
    ki=0.33
    ks=1-ko-ki
    dns = 10 # Porcentaje de demanda no satisfecha que podemos asumir
    #DECLARAMOS VARIABLES DE LAS DISTRIBUCIONES NORMALES
    mu = 100
    sigma = 30
    beta = 0.5
    xi = 0.05
    #-----------------------------------------------------
    #VARIABLES VECTORIALES, CAMBIAN DE VALOR EN CADA ITERACIÓN
    d = []  #Demanda real
    r = []  #Devoluciones
    pd = [] #Previsión demanda
    ss = [] #Stock de seguridad
    tw = [] #Trabajo en curso objetivo
    mc = [] #Unidades nuevas que finalizan en t
    rc = [] #Unidades refabricadas que finalizan en t
    si = [] #Stock inicial en t
    ns = [] #Stock neto en t
    o = []  #Orden de producción en t
    w = []  #Trabajo en curso en t
    j = []  #resultado de cada ejecución
    #INICIO DE SIMULACIÓN
    e = 1
    while e<E:
        t = 0
        d.clear()
        r.clear()
        pd.clear()
```

```
        ss.clear()
        tw.clear()
        mc.clear()
        rc.clear()
        si.clear()
        ns.clear()
        o.clear()
        w.clear()
        while t<T:
            #DEMANDA DEL CLIENTE t
            xt = np.random.normal(loc = mu, scale = sigma, size = None)
            if xt > 0:
                d.append(xt)
            else:
                d.append(0)
            #DEVOLUCIONES DEL CLIENTE EN t
            zt = np.random.normal(loc = beta, scale = xi, size = None)
            if zt > 0 and zt < 1:
                yt = zt
            elif zt < 0:
                yt = 0
            else:
                yt = 1
            if t>=tc:
                r.append(yt*d[t-tc])
            else:
                r.append(0)
            #PREVISIÓN DE LA DEMANDA EN t
            if t>0:
                pd.append((alpha*d[t])+(1-alpha)*pd[t-1])
            else:
                pd.append(alpha*d[t])
            #STOCK DE SEGURIDAD EN t
            ss.append(epsilon*pd[t])
            #TRABAJO EN CURSO OBJETIVO EN t
            tw.append(tp*pd[t])
            #UNIDADES NUEVAS QUE FINALIZAN EN EL PERIODO t
            if t-tm-1 < 0:
                mc.append(0)
            else:
                mc.append(o[t-tm-1])
            #UNIDADES REFABRICADAS QUE FINALIZAN EN EL PERIODO t
            if t-tr-1 < 0:
                rc.append(0)
            else:
                rc.append(r[t-tr-1])
            #STOCK INICIAL EN t
            if t<1:
                si.append(0)
            else:
```

```
                si.append(ns[t-1]+mc[t]+rc[t])
            #STOCK NETO EN t
            ns.append(si[t]-d[t])
            #TRABAJO EN CURSO EN t
            if t<1:
                w.append(0)
            else:
                w.append(w[t-1]+(o[t-1]-mc[t])+(r[t-1]-rc[t]))
            #ORDEN DE PRODUCCIÓN EN t
            aux = ((pd[t]-rc[t])+(1/Ti)*(ss[t]-ns[t])+(1/Tw)*(tw[t]-w[t])
                )
            if aux > 0:
                o.append(aux)
            else:
                o.append(0)
            t = t+1

        #VALIDACIÓN DE LA SOLUCIÓN
        neg = 0
        for num in ns[99:]:
            if num < 0:
                neg += 1
        porcneg=(neg/2000)*100
        if porcneg < dns:
            #CÁLCULO PARÁMETROS DE RENDIMIENTO
            varo = var(o[99:])
            vard = var(d[99:])
            varns = var(ns[99:])
            OVR = varo/vard
            IVR = varns/vard
            nsm = mean(ns[99:])
            j.append((ko*np.sqrt(OVR))+(ki*np.sqrt(IVR))+(ks*(nsm/100)))
        e = e+1
    if len(j)<9:
        J = 10000000
    else:
        J = mean(j)
    return J

#------------------Variante 2--------------------------------
import random
    Jmin =10000000
    while Jmin==10000000:
        Ti=random.randint(1,64)
        Tp=random.randint(0,6)
        alpha=random.uniform(0,1)
        epsilon=random.uniform(0,3)
        Jmin=evaluarJ2(Ti,Ti,Tp,alpha,epsilon)
    eliminarA=5
    eliminarB=5
```

```
nomejora=0
while nomejora < 90:
    mejora=0
    eliminarantA=eliminarA
    eliminarantB=eliminarB
    while eliminarA==eliminarantA or eliminarA == eliminarantB : #
        Forzamos que al menos uno de los valores cambie
        eliminarA=random.randint(0,3)
    eliminarB=random.randint(0,3)
    while (eliminarA==eliminarB):
        eliminarB=random.randint(0,3)
 #---------------6 Variaciones posibles---------------------


#--------------------VARIACIÓN 1------------
   #Aproximación inicial +-5
    if ((eliminarA==0 and eliminarB==1)or(eliminarA==1 and eliminarB
        ==0)):
        Tpaux=0.5
        while Tpaux<6:
            Tiaux=1
            while Tiaux < 64:
                J=evaluarJ2(Tiaux,Tiaux,Tpaux,alpha,epsilon)
                if J<Jmin:
                    print("Var 1 simple",J)
                    Jmin = J
                    Ti=Tiaux
                    Tp=Tpaux
                    mejora=1
                Tiaux = Tiaux + 5
            Tpaux = Tpaux + 0.25

        if mejora == 1:
            #Aproximación final +-1 (caso A)
            if Ti-10>0 and Ti+10<64:
                Tiaux=Ti-10
                CotasupTi=Ti+10
                while Tiaux<CotasupTi:
                     J=evaluarJ2(Tiaux,Tiaux,Tp,alpha,epsilon)
                     if J<Jmin:
                         print("Var 1 bucle A",J)
                         Jmin = J
                         Ti=Tiaux
                     Tiaux = Tiaux + 1
            #Aproximación final +-1 (caso B)
            elif Ti-10<0:
                Tiaux=1
                CotasupTi=Ti+10
                while Tiaux<CotasupTi:
                     J=evaluarJ2(Tiaux,Tiaux,Tp,alpha,epsilon)
```

```
                    if J<Jmin:
                        print("Var 1 bucle B",J)
                        Jmin = J
                        Ti=Tiaux
                    Tiaux = Tiaux + 1
            #Aproximación final +-1 (caso C)
            elif Ti+10>64:
                Tiaux=Ti-10
                CotasupTi=64
                while Tiaux<CotasupTi:
                    J=evaluarJ2(Tiaux,Tiaux,Tp,alpha,epsilon)
                    if J<Jmin:
                        print("Var 2 bucle C",J)
                        Jmin = J
                        Ti=Tiaux
                    Tiaux = Tiaux + 1

#------------VARIACIÓN 2-----------------------
   #Aproximación inicial +-5
    elif ((eliminarA==0 and eliminarB==2)or(eliminarA==2 and
        eliminarB==0)):
        alphaux=0
        while alphaux<1:
            Tiaux=1
            while Tiaux < 64:
                J=evaluarJ2(Tiaux,Tiaux,Tp,alphaux,epsilon)
                if J<Jmin:
                    print("Var 2 simple",J)
                    Jmin = J
                    Ti=Tiaux
                    alpha=alphaux
                    mejora=1
                Tiaux = Tiaux + 5
            alphaux = alphaux + 0.1
        if mejora == 1:
            #Aproximación final +-1 (caso A)
            if Ti-10>0 and Ti+10<64:
                Tiaux=Ti-10
                CotasupTi=Ti+10
                while Tiaux<CotasupTi:
                    J=evaluarJ2(Tiaux,Tiaux,Tp,alpha,epsilon)
                    if J<Jmin:
                        print("Var 2 bucle A",J)
                        Jmin = J
                        Ti=Tiaux
                    Tiaux = Tiaux + 1
            #Aproximación final +-1 (caso B)
            elif Ti-10<0:
                Tiaux=1
                CotasupTi=Ti+10
```

```
                while Tiaux<CotasupTi:
                    J=evaluarJ2(Tiaux,Tiaux,Tp,alpha,epsilon)
                    if J<Jmin:
                        print("Var 2 bucle B",J)
                        Jmin = J
                        Ti=Tiaux
                    Tiaux = Tiaux + 1
            #Aproximación final +-1 (caso C)
            elif Ti+10>64:
                Tiaux=Ti-10
                CotasupTi=64
                while Tiaux<CotasupTi:
                    J=evaluarJ2(Tiaux,Tiaux,Tp,alpha,epsilon)
                    if J<Jmin:
                        print("Var 2 bucle C",J)
                        Jmin = J
                        Ti=Tiaux
                    Tiaux = Tiaux + 1

#----------------VARIACIÓN 3----------------
   #Aproximación inicial +-5
    elif ((eliminarA==0 and eliminarB==3)or(eliminarA==3 and
        eliminarB==0)):
        epsilonaux=0
        while epsilonaux<3:
            Tiaux=1
            while Tiaux < 64:
                J=evaluarJ2(Tiaux,Tiaux,Tp,alpha,epsilonaux)
                if J<Jmin:
                    print("Var 4 simple",J)
                    Jmin = J
                    Ti=Tiaux
                    epsilon=epsilonaux
                    mejora=1
                Tiaux = Tiaux + 5
            epsilonaux = epsilonaux + 0.1
        if mejora == 1:
            #Aproximación final +-1 (caso A)
            if Ti-10>0 and Ti+10<64:
                Tiaux=Ti-10
                CotasupTi=Ti+10
                while Tiaux<CotasupTi:
                    J=evaluarJ2(Tiaux,Tiaux,Tp,alpha,epsilon)
                    if J<Jmin:
                        print("Var 4 bucle A",J)
                        Jmin = J
                        Ti=Tiaux
                    Tiaux = Tiaux + 1
            #Aproximación final +-1 (caso B)
            elif Ti-10<0:
```

```
                Tiaux=1
                CotasupTi=Ti+10
                while Tiaux<CotasupTi:
                    J=evaluarJ2(Tiaux,Tiaux,Tp,alpha,epsilon)
                    if J<Jmin:
                        print("Var 4 bucle B",J)
                        Jmin = J
                        Ti=Tiaux
                    Tiaux = Tiaux + 1
            #Aproximación final +-1 (caso C)
            elif Ti+10>64:
                Tiaux=Ti-10
                CotasupTi=64
                while Tiaux<CotasupTi:
                    J=evaluarJ2(Tiaux,Tiaux,Tp,alpha,epsilon)
                    if J<Jmin:
                        print("Var 4 bucle C",J)
                        Jmin = J
                        Ti=Tiaux
                    Tiaux = Tiaux + 1



#-------------------VARIACIÓN 4 ----------------
    elif ((eliminarA==1 and eliminarB==2)or(eliminarA==2 and
        eliminarB==1)):
        Tpaux=0.5
        while Tpaux<6:
            alphaux=0
            while alphaux < 1:
                J=evaluarJ2(Ti,Ti,Tpaux,alphaux,epsilon)
                if J<Jmin:
                    print("Var 4",J)
                    Jmin = J
                    alpha = alphaux
                    Tp=Tpaux
                    mejora=1
                alphaux = alphaux + 0.1
            Tpaux = Tpaux + 0.25

#-----------------VARIACIÓN 5------------------
    elif ((eliminarA==1 and eliminarB==3)or(eliminarA==3 and
        eliminarB==1)):
        Tpaux=0.5
        while Tpaux<6:
            epsilonaux=0
            while epsilonaux < 3:
                J=evaluarJ2(Ti,Ti,Tpaux,alpha,epsilonaux)
                if J<Jmin:
                    print("Var 5",J)
```

```
                    Jmin = J
                    epsilon=epsilonaux
                    Tp=Tpaux
                    mejora=1
                epsilonaux = epsilonaux + 0.1
            Tpaux = Tpaux + 0.25

#--------------VARIACIÓN 6--------------------
    elif ((eliminarA==2 and eliminarB==3)or(eliminarA==3 and
        eliminarB==2)):
        alphaux=0
        while alphaux<1:
            epsilonaux=0
            while epsilonaux < 3:
                J=evaluarJ2(Ti,Ti,Tp,alphaux,epsilonaux)
                if J<Jmin:
                    print("Var 6",J)
                    Jmin = J
                    epsilon=epsilonaux
                    alpha=alphaux
                    mejora=1
                epsilonaux = epsilonaux + 0.1
            alphaux = alphaux + 0.1

    if mejora == 0:
        nomejora = nomejora + 1
    elif mejora == 1:
       nomejora = 0
print("Ti=",Ti,"Tw=",Ti,"Tp=",Tp,"alpha",alpha,"Epsilon",epsilon)
```

ANEXO II

1 Códigos

Código 5.1 Variante 1 original (Fuente: Elaboración propia).

```
import random
Jmin =10000000
while Jmin==10000000:
    Ti=random.randint(1,64)
    Tw=random.randint(1,64)
    Tp=random.uniform(0.5,6)
    Jmin=evaluarJmod(Ti,Tw,Tp)
eliminar=5
nomejora=0
while nomejora < 300:
    mejora=0
    eliminarant=eliminar
    while eliminarant==eliminar:
        eliminar=random.randint(0,2)
    if eliminar== 0:
        Tiaux=1
        while Tiaux < 62:
            J=evaluarJmod(Tiaux,Tw,Tp)
            if J<Jmin:
                print("bucle 1",J)
                Jmin = J
                Ti=Tiaux
                mejora=1
            Tiaux = Tiaux + 5
        if mejora == 1:
            if Ti-10>0 and Ti+10<64:
                Tiaux=Ti-10
                Cotasup=Ti+10
                while Tiaux<Cotasup:
```

```
                J=evaluarJmod(Tiaux,Tw,Tp)
                if J<Jmin:
                    print("bucle 1 a",J)
                    Jmin = J
                    Ti=Tiaux
                Tiaux = Tiaux + 1
        elif Ti-10<0:
            Tiaux=1
            Cotasup=Ti+10
            while Tiaux<Cotasup:
                J=evaluarJmod(Tiaux,Tw,Tp)
                if J<Jmin:
                    print("bucle 1 b",J)
                    Jmin = J
                    Ti=Tiaux
                Tiaux = Tiaux + 1
        elif Ti+10>64:
            Tiaux=Ti-10
            Cotasup=64
            while Tiaux<Cotasup:
                J=evaluarJmod(Tiaux,Tw,Tp)
                if J<Jmin:
                    print("bucle 1 c",J)
                    Jmin = J
                    Ti=Tiaux
                Tiaux = Tiaux + 1
    if eliminar== 1:
        Twaux=1
        while Twaux < 62:
            J=evaluarJmod(Ti,Twaux,Tp)
            if J<Jmin:
                print("bucle 2",J)
                Jmin = J
                Tw=Twaux
                mejora=1
            Twaux = Twaux + 5
        if mejora == 1:
            if Tw-10>0 and Tw+10<64:
                Twaux=Tw-10
                Cotasup=Tw+10
                while Twaux<Cotasup:
                    J=evaluarJmod(Ti,Twaux,Tp)
                    if J<Jmin:
                        print("bucle 2 a",J)
                        Jmin = J
                        Tw=Twaux
                    Twaux = Twaux + 1
            elif Tw-10<0:
                Twaux=1
                Cotasup=Tw+10
```

```
                while Twaux<Cotasup:
                    J=evaluarJmod(Ti,Twaux,Tp)
                    if J<Jmin:
                        print("bucle 2 b",J)
                        Jmin = J
                        Tw=Twaux
                    Twaux = Twaux + 1
            elif Tw+10>64:
                Twaux=Tw-10
                Cotasup=64
                while Twaux<Cotasup:
                    J=evaluarJmod(Ti,Twaux,Tp)
                    if J<Jmin:
                        print("bucle 2 c",J)
                        Jmin = J
                        Tw=Twaux
                    Twaux = Twaux + 1

    elif eliminar== 2:
        Tpaux=0.5
        while Tpaux < 6:
            J=evaluarJmod(Ti,Tw,Tpaux)
            if J<Jmin:
                print("bucle 3",J)
                Jmin = J
                Tp=Tpaux
                mejora=1
            Tpaux = Tpaux + 0.25
    if mejora == 0:
        nomejora = nomejora + 1
    elif mejora == 1:
       nomejora = 0
print("Ti=",Ti,"Tw=",Tw,"Tp=",Tp)
Japrox = evaluarJ(Ti,Tw,Tp)
print("Japrox",Japrox)
```

Código 5.2 Variante 2 original (Fuente: Elaboración propia).

```
import random
Jmin =10000000
while Jmin==10000000:
    Ti=random.randint(1,64)
    Tw=random.randint(1,64)
    Tp=random.randint(0,6)
    alpha=random.uniform(0,1)
    epsilon=random.uniform(0,3)
    Jmin=evaluarJ2(Ti,Tw,Tp,alpha,epsilon)
eliminarA=5
eliminarB=5
```

```
nomejora=0
while nomejora < 30:
    mejora=0
    eliminarantA=eliminarA
    eliminarantB=eliminarB
    while eliminarA==eliminarantA or eliminarA == eliminarantB : #
        Forzamos que al menos uno de los valores cambie
        eliminarA=random.randint(0,4)
    eliminarB=random.randint(0,4)
    while (eliminarA==eliminarB):
        eliminarB=random.randint(0,4)
 #------------------10 Variaciones posibles----------------------
 #--------------VARIACIÓN 1------------
    #Aproximación inicial +-5
    if ((eliminarA==0 and eliminarB==1)or(eliminarA==1 and eliminarB==0)
        ):
        Tiaux=1
        while Tiaux < 62:
            Twaux = 1
            while Twaux < 62:
                J=evaluarJ2(Tiaux,Twaux,Tp,alpha,epsilon)
                if J<Jmin:
                    print("Var 1 simple",J)
                    Jmin = J
                    Ti=Tiaux
                    Tw=Twaux
                    mejora=1
                Twaux = Twaux + 5
            Tiaux = Tiaux + 5

        if mejora == 1:
             #Aproximación final +-1 (caso A)
            if ((Ti-10>0 and Ti+10<64)and(Tw-10>0 and Tw+10<64)):
                Tiaux=Ti-10
                CotasupTi=Ti+10
                CotasupTw=Tw+10
                while Tiaux<CotasupTi:
                    Twaux=Tw-10
                    while Twaux<CotasupTw:
                        J=evaluarJ2(Tiaux,Twaux,Tp,alpha,epsilon)
                        if J<Jmin:
                            print("Var 1 bucle A",J)
                            Jmin = J
                            Ti=Tiaux
                            Tw=Twaux
                        Twaux = Twaux + 1
                    Tiaux = Tiaux + 1

            #Aproximación final +-1 (caso B)
            elif ((Ti-10<0) and (Tw-10>0 and Tw+10<64)):
```

```
            Tiaux=1
            CotasupTi=Ti+10
            CotasupTw=Tw+10
            while Tiaux<CotasupTi:
                Twaux=Tw-10
                while Twaux<CotasupTw:
                    J=evaluarJ2(Tiaux,Twaux,Tp,alpha,epsilon)
                    if J<Jmin:
                        print("Var 1 bucle B",J)
                        Jmin = J
                        Ti=Tiaux
                        Tw=Twaux
                    Twaux = Twaux + 1
                Tiaux = Tiaux + 1
        #Aproximación final +-1 (caso C)
        elif ((Ti+10>64) and (Tw-10>0 and Tw+10<64)):
            Tiaux=Ti-10
            CotasupTi=64
            CotasupTw=Tw+10
            while Tiaux<CotasupTi:
                Twaux=Tw-10
                while Twaux<CotasupTw:
                    J=evaluarJ2(Tiaux,Twaux,Tp,alpha,epsilon)
                    if J<Jmin:
                        print("Var 1 bucle C",J)
                        Jmin = J
                        Ti=Tiaux
                        Tw=Twaux
                    Twaux = Twaux + 1
                Tiaux = Tiaux + 1

        #Aproximación final +-1 (caso D)
        elif ((Ti-10>0 and Ti+10<64)and(Tw-10<0)):
            Tiaux=Ti-10
            CotasupTi=Ti+10
            CotasupTw=Tw+10
            while Tiaux<CotasupTi:
                Twaux=1
                while Twaux<CotasupTw:
                    J=evaluarJ2(Tiaux,Twaux,Tp,alpha,epsilon)
                    if J<Jmin:
                        print("Var 1 bucle D",J)
                        Jmin = J
                        Ti=Tiaux
                        Tw=Twaux
                    Twaux = Twaux + 1
                Tiaux = Tiaux + 1

        #Aproximación final +-1 (caso E)
        elif ((Ti-10>0 and Ti+10<64)and(Tw+10>64)):
```

```
                Tiaux=Ti-10
                CotasupTi=Ti+10
                CotasupTw=64
                while Tiaux<CotasupTi:
                    Twaux=Tw-10
                    while Twaux<CotasupTw:
                        J=evaluarJ2(Tiaux,Twaux,Tp,alpha,epsilon)
                        if J<Jmin:
                            print("Var 1 bucle E",J)
                            Jmin = J
                            Ti=Tiaux
                            Tw=Twaux
                        Twaux = Twaux + 1
                    Tiaux = Tiaux + 1
            #Aproximación final +-1 (caso F)
            elif ((Ti-10<0) and (Tw-10<0)):
                Tiaux=1
                CotasupTi=Ti+10
                CotasupTw=Tw+10
                while Tiaux<CotasupTi:
                    Twaux=1
                    while Twaux<CotasupTw:
                        J=evaluarJ2(Tiaux,Twaux,Tp,alpha,epsilon)
                        if J<Jmin:
                            print("Var 1 bucle F",J)
                            Jmin = J
                            Ti=Tiaux
                            Tw=Twaux
                        Twaux = Twaux + 1
                    Tiaux = Tiaux + 1
            #Aproximación final +-1 (caso G)
            elif ((Ti-10<0) and (Tw+10>64)):
                Tiaux=1
                CotasupTi=Ti+10
                CotasupTw=64
                while Tiaux<CotasupTi:
                    Twaux=Tw-10
                    while Twaux<CotasupTw:
                        J=evaluarJ2(Tiaux,Twaux,Tp,alpha,epsilon)
                        if J<Jmin:
                            print("Var 1 bucle G",J)
                            Jmin = J
                            Ti=Tiaux
                            Tw=Twaux
                        Twaux = Twaux + 1
                    Tiaux = Tiaux + 1
            #Aproximación final +-1 (caso H)
            elif ((Ti+10>64) and (Tw-10<0)):
                Tiaux=Ti-10
                CotasupTi=64
```

```
                CotasupTw=Tw+10
                while Tiaux<CotasupTi:
                    Twaux=1
                    while Twaux<CotasupTw:
                        J=evaluarJ2(Tiaux,Twaux,Tp,alpha,epsilon)
                        if J<Jmin:
                            print("Var 1 bucle H",J)
                            Jmin = J
                            Ti=Tiaux
                            Tw=Twaux
                        Twaux = Twaux + 1
                    Tiaux = Tiaux + 1
            #Aproximación final +-1 (caso I)
            elif ((Ti+10>64) and (Tw+10>64)):
                Tiaux=Ti-10
                CotasupTi=64
                CotasupTw=64
                while Tiaux<CotasupTi:
                    Twaux=Tw-10
                    while Twaux<CotasupTw:
                        J=evaluarJ2(Tiaux,Twaux,Tp,alpha,epsilon)
                        if J<Jmin:
                            print("Var 1 bucle I",J)
                            Jmin = J
                            Ti=Tiaux
                            Tw=Twaux
                        Twaux = Twaux + 1
                    Tiaux = Tiaux + 1


#-------------VARIACIÓN 2------------
   #Aproximación inicial +-5
    elif ((eliminarA==0 and eliminarB==2)or(eliminarA==2 and eliminarB
        ==0)):
        Tpaux=0.5
        while Tpaux<6:
            Tiaux=1
            while Tiaux < 64:
                J=evaluarJ2(Tiaux,Tw,Tpaux,alpha,epsilon)
                if J<Jmin:
                    print("Var 2 simple",J)
                    Jmin = J
                    Ti=Tiaux
                    Tp=Tpaux
                    mejora=1
                Tiaux = Tiaux + 5
            Tpaux = Tpaux + 0.25

        if mejora == 1:
            #Aproximación final +-1 (caso A)
```

```
            if Ti-10>0 and Ti+10<64:
                Tiaux=Ti-10
                CotasupTi=Ti+10
                while Tiaux<CotasupTi:
                     J=evaluarJ2(Tiaux,Tw,Tp,alpha,epsilon)
                     if J<Jmin:
                         print("Var 2 bucle A",J)
                         Jmin = J
                         Ti=Tiaux
                     Tiaux = Tiaux + 1
            #Aproximación final +-1 (caso B)
            elif Ti-10<0:
                Tiaux=1
                CotasupTi=Ti+10
                while Tiaux<CotasupTi:
                     J=evaluarJ2(Tiaux,Tw,Tp,alpha,epsilon)
                     if J<Jmin:
                         print("Var 2 bucle B",J)
                         Jmin = J
                         Ti=Tiaux
                     Tiaux = Tiaux + 1
            #Aproximación final +-1 (caso C)
            elif Ti+10>64:
                Tiaux=Ti-10
                CotasupTi=64
                while Tiaux<CotasupTi:
                     J=evaluarJ2(Tiaux,Tw,Tp,alpha,epsilon)
                     if J<Jmin:
                         print("Var 2 bucle C",J)
                         Jmin = J
                         Ti=Tiaux
                     Tiaux = Tiaux + 1

#----------------VARIACIÓN 3---------------
   #Aproximación inicial +-5
    elif ((eliminarA==0 and eliminarB==3)or(eliminarA==3 and eliminarB
        ==0)):
        alphaux=0
        while alphaux<1:
            Tiaux=1
            while Tiaux < 64:
                J=evaluarJ2(Tiaux,Tw,Tp,alphaux,epsilon)
                if J<Jmin:
                    print("Var 3 simple",J)
                    Jmin = J
                    Ti=Tiaux
                    alpha=alphaux
                    mejora=1
                Tiaux = Tiaux + 5
            alphaux = alphaux + 0.1
```

```
        if mejora == 1:
            #Aproximación final +-1 (caso A)
            if Ti-10>0 and Ti+10<64:
                Tiaux=Ti-10
                CotasupTi=Ti+10
                while Tiaux<CotasupTi:
                     J=evaluarJ2(Tiaux,Tw,Tp,alpha,epsilon)
                     if J<Jmin:
                         print("Var 3 bucle A",J)
                         Jmin = J
                         Ti=Tiaux
                     Tiaux = Tiaux + 1
            #Aproximación final +-1 (caso B)
            elif Ti-10<0:
                Tiaux=1
                CotasupTi=Ti+10
                while Tiaux<CotasupTi:
                     J=evaluarJ2(Tiaux,Tw,Tp,alpha,epsilon)
                     if J<Jmin:
                         print("Var 3 bucle B",J)
                         Jmin = J
                         Ti=Tiaux
                     Tiaux = Tiaux + 1
            #Aproximación final +-1 (caso C)
            elif Ti+10>64:
                Tiaux=Ti-10
                CotasupTi=64
                while Tiaux<CotasupTi:
                     J=evaluarJ2(Tiaux,Tw,Tp,alpha,epsilon)
                     if J<Jmin:
                         print("Var 3 bucle C",J)
                         Jmin = J
                         Ti=Tiaux
                     Tiaux = Tiaux + 1

#-------------VARIACIÓN 4--------------
   #Aproximación inicial +-5
    elif ((eliminarA==0 and eliminarB==4)or(eliminarA==4 and eliminarB
        ==0)):
        epsilonaux=0
        while epsilonaux<3:
            Tiaux=1
            while Tiaux < 64:
                J=evaluarJ2(Tiaux,Tw,Tp,alpha,epsilonaux)
                if J<Jmin:
                    print("Var 4 simple",J)
                    Jmin = J
                    Ti=Tiaux
                    epsilon=epsilonaux
                    mejora=1
```

```
                Tiaux = Tiaux + 5
            epsilonaux = epsilonaux + 0.1
        if mejora == 1:
            #Aproximación final +-1 (caso A)
            if Ti-10>0 and Ti+10<64:
                Tiaux=Ti-10
                CotasupTi=Ti+10
                while Tiaux<CotasupTi:
                    J=evaluarJ2(Tiaux,Tw,Tp,alpha,epsilon)
                    if J<Jmin:
                        print("Var 4 bucle A",J)
                        Jmin = J
                        Ti=Tiaux
                    Tiaux = Tiaux + 1
            #Aproximación final +-1 (caso B)
            elif Ti-10<0:
                Tiaux=1
                CotasupTi=Ti+10
                while Tiaux<CotasupTi:
                    J=evaluarJ2(Tiaux,Tw,Tp,alpha,epsilon)
                    if J<Jmin:
                        print("Var 4 bucle B",J)
                        Jmin = J
                        Ti=Tiaux
                    Tiaux = Tiaux + 1
            #Aproximación final +-1 (caso C)
            elif Ti+10>64:
                Tiaux=Ti-10
                CotasupTi=64
                while Tiaux<CotasupTi:
                    J=evaluarJ2(Tiaux,Tw,Tp,alpha,epsilon)
                    if J<Jmin:
                        print("Var 4 bucle C",J)
                        Jmin = J
                        Ti=Tiaux
                    Tiaux = Tiaux + 1

#----------------VARIACIÓN 5---------------
   #Aproximación inicial +-5
    elif ((eliminarA==1 and eliminarB==2)or(eliminarA==2 and eliminarB
        ==1)):
        Tpaux=0.5
        while Tpaux<6:
            Twaux=1
            while Twaux < 64:
                J=evaluarJ2(Ti,Twaux,Tpaux,alpha,epsilon)
                if J<Jmin:
                    print("Var 5 simple",J)
                    Jmin = J
                    Tw=Twaux
```

```
                    Tp=Tpaux
                    mejora=1
                Twaux = Twaux + 5
            Tpaux = Tpaux + 0.25
        if mejora == 1:
            #Aproximación final +-1 (caso A)
            if Tw-10>0 and Tw+10<64:
                Twaux=Tw-10
                CotasupTw=Tw+10
                while Twaux<CotasupTw:
                     J=evaluarJ2(Ti,Twaux,Tp,alpha,epsilon)
                     if J<Jmin:
                         print("Var 5 bucle A",J)
                         Jmin = J
                         Tw=Twaux
                     Twaux = Twaux + 1
            #Aproximación final +-1 (caso B)
            elif Tw-10<0:
                Twaux=1
                CotasupTw=Tw+10
                while Twaux<CotasupTw:
                     J=evaluarJ2(Ti,Twaux,Tp,alpha,epsilon)
                     if J<Jmin:
                         print("Var 5 bucle B",J)
                         Jmin = J
                         Tw=Twaux
                     Twaux = Twaux + 1
            #Aproximación final +-1 (caso C)
            elif Tw+10>64:
                Twaux=Tw-10
                CotasupTw=64
                while Twaux<CotasupTw:
                     J=evaluarJ2(Ti,Twaux,Tp,alpha,epsilon)
                     if J<Jmin:
                         print("Var 5 bucle C",J)
                         Jmin = J
                         Tw=Twaux
                     Twaux = Twaux + 1

#---------------VARIACIÓN 6---------------
   #Aproximación inicial +-5
    elif ((eliminarA==1 and eliminarB==3)or(eliminarA==3 and eliminarB
        ==1)):
        alphaux=0
        while alphaux<1:
            Twaux=1
            while Twaux < 64:
                J=evaluarJ2(Ti,Twaux,Tp,alphaux,epsilon)
                if J<Jmin:
                    print("Var 6 simple",J)
```

```
                    Jmin = J
                    Tw=Twaux
                    alpha=alphaux
                    mejora=1
                Twaux = Twaux + 5
            alphaux = alphaux + 0.1
        if mejora == 1:
            #Aproximación final +-1 (caso A)
            if Tw-10>0 and Tw+10<64:
                Twaux=Tw-10
                CotasupTw=Tw+10
                while Twaux<CotasupTw:
                    J=evaluarJ2(Ti,Twaux,Tp,alpha,epsilon)
                    if J<Jmin:
                        print("Var 6 bucle A",J)
                        Jmin = J
                        Tw=Twaux
                    Twaux = Twaux + 1
            #Aproximación final +-1 (caso B)
            elif Tw-10<0:
                Twaux=1
                CotasupTw=Tw+10
                while Twaux<CotasupTw:
                    J=evaluarJ2(Ti,Twaux,Tp,alpha,epsilon)
                    if J<Jmin:
                        print("Var 6 bucle B",J)
                        Jmin = J
                        Tw=Twaux
                    Twaux = Twaux + 1
            #Aproximación final +-1 (caso C)
            elif Tw+10>64:
                Twaux=Tw-10
                CotasupTw=64
                while Twaux<CotasupTw:
                    J=evaluarJ2(Ti,Twaux,Tp,alpha,epsilon)
                    if J<Jmin:
                        print("Var 6 bucle C",J)
                        Jmin = J
                        Tw=Twaux
                    Twaux = Twaux + 1

#---------------VARIACIÓN 7---------------
   #Aproximación inicial +-5
    elif ((eliminarA==1 and eliminarB==4)or(eliminarA==4 and eliminarB
        ==1)):
        epsilonaux=0
        while epsilonaux<1:
            Twaux=1
            while Twaux < 64:
                J=evaluarJ2(Ti,Twaux,Tp,alpha,epsilonaux)
```

```
                if J<Jmin:
                    print("Var 7 simple",J)
                    Jmin = J
                    Tw=Twaux
                    epsilon=epsilonaux
                    mejora=1
                Twaux = Twaux + 5
            epsilonaux = epsilonaux + 0.1
        if mejora == 1:
            #Aproximación final +-1 (caso A)
            if Tw-10>0 and Tw+10<64:
                Twaux=Tw-10
                CotasupTw=Tw+10
                while Twaux<CotasupTw:
                     J=evaluarJ2(Ti,Twaux,Tp,alpha,epsilon)
                     if J<Jmin:
                         print("Var 7 bucle A",J)
                         Jmin = J
                         Tw=Twaux
                     Twaux = Twaux + 1
            #Aproximación final +-1 (caso B)
            elif Tw-10<0:
                Twaux=1
                CotasupTw=Tw+10
                while Twaux<CotasupTw:
                     J=evaluarJ2(Ti,Twaux,Tp,alpha,epsilon)
                     if J<Jmin:
                         print("Var 7 bucle B",J)
                         Jmin = J
                         Tw=Twaux
                     Twaux = Twaux + 1
            #Aproximación final +-1 (caso C)
            elif Tw+10>64:
                Twaux=Tw-10
                CotasupTw=64
                while Twaux<CotasupTw:
                     J=evaluarJ2(Ti,Twaux,Tp,alpha,epsilon)
                     if J<Jmin:
                         print("Var 7 bucle C",J)
                         Jmin = J
                         Tw=Twaux
                     Twaux = Twaux + 1

#----------------VARIACIÓN 8--------------
    elif ((eliminarA==2 and eliminarB==3)or(eliminarA==3 and eliminarB
        ==2)):
        Tpaux=0.5
        while Tpaux<6:
            alphaux=0
            while alphaux < 1:
```

```
                J=evaluarJ2(Ti,Tw,Tpaux,alphaux,epsilon)
                if J<Jmin:
                    print("Var 8",J)
                    Jmin = J
                    alpha = alphaux
                    Tp=Tpaux
                    mejora=1
                alphaux = alphaux + 0.1
            Tpaux = Tpaux + 0.25

#-----------------VARIACIÓN 9----------------
    elif ((eliminarA==2 and eliminarB==4)or(eliminarA==4 and eliminarB
        ==2)):
        Tpaux=0.5
        while Tpaux<6:
            epsilonaux=0
            while epsilonaux < 3:
                J=evaluarJ2(Ti,Tw,Tpaux,alpha,epsilonaux)
                if J<Jmin:
                    print("Var 9",J)
                    Jmin = J
                    epsilon=epsilonaux
                    Tp=Tpaux
                    mejora=1
                epsilonaux = epsilonaux + 0.1
            Tpaux = Tpaux + 0.25

#----------------VARIACIÓN 10----------------
    elif ((eliminarA==3 and eliminarB==4)or(eliminarA==4 and eliminarB
        ==3)):
        alphaux=0
        while alphaux<1:
            epsilonaux=0
            while epsilonaux < 3:
                J=evaluarJ2(Ti,Tw,Tp,alphaux,epsilonaux)
                if J<Jmin:
                    print("Var 10",J)
                    Jmin = J
                    epsilon=epsilonaux
                    alpha=alphaux
                    mejora=1
                epsilonaux = epsilonaux + 0.1
            alphaux = alphaux + 0.1

    if mejora == 0:
        nomejora = nomejora + 1
    elif mejora == 1:
       nomejora = 0
print("Ti=",Ti,"Tw=",Tw,"Tp=",Tp,"alpha",alpha,"Epsilon",epsilon)
```

2 Tablas de resultados

Tabla 1 Resultados variante 1 original. α=0.1, ε=1, β= 0.5 (Fuente: elaboración propia).

K_o	K_i	Nº NO mejora	T_i	T_w	T_p	J mod	J aprox
0.7	0.15	300	31	31	3,25	1,064	1,945
0.7	0.15	300	46	41	3,25	1,068	1,943
0.7	0.15	300	36	61	3,5	1,073	1,883
0.7	0.15	300	31	31	3,25	1,067	1,864
0.7	0.15	300	26	24	3,25	1,073	1,869
0.7	0.15	300	41	61	3,5	1,078	1,885
0.7	0.15	300	36	61	3,5	1,074	1,903
0.7	0.15	300	41	56	3,5	1,078	1,887
0.7	0.15	300	46	62	3,5	1,076	1,947
0.7	0.15	300	31	61	3,5	1,071	1,9
0.15	0.7	300	36	61	3,5	2,406	1,868
0.15	0.7	300	11	11	3,25	2,376	1,906
0.15	0.7	300	36	61	3,75	2,443	1,883
0.15	0.7	300	26	61	3,5	2,405	1,899
0.15	0.7	300	31	61	3,5	2,414	1,893
0.15	0.7	300	26	56	4	2,462	1,948
0.15	0.7	300	5	3	3,25	2,304	1,858
0.15	0.7	300	31	56	3,5	2,403	1,856
0.15	0.7	300	15	23	3,5	2,393	1,927
0.15	0.7	300	28	46	3,5	2,419	1,921
0.15	0.15	300	31	62	3,5	1,413	1,892
0.15	0.15	300	16	11	3,127	1,376	1,903
0.15	0.15	300	11	61	4,5	1,491	2,074
0.15	0.15	300	21	26	3,25	1,391	1,891
0.15	0.15	300	4	16	4,25	1,592	2,2
0.15	0.15	300	21	63	3,75	1,438	1,969
0.15	0.15	300	26	31	3,25	1,4	1,942
0.15	0.15	300	10	63	5	1,53	2,073
0.15	0.15	300	6	6	3,25	1,436	1,911
0.15	0.15	300	7	51	5	1,538	2,155

Tabla 2 Resultados variante 2 original. β= 0.5 (Fuente: elaboración propia).

K_o	K_i	Nº NO mejora	T_i	T_w	T_p	α	ε	J mod	J aprox
0.7	0.15	30	50	41	1,75	0,2	2,67	1,05	1,767
0.7	0.15	30	60	59	3,25	0,2	0,931	1,061	1,821
0.7	0.15	30	49	16	2,75	0,2	2	1,059	1,814
0.7	0.15	30	41	11	3	0,2	1,148	1,051	1,749
0.7	0.15	30	38	7	2,75	0,2	2,5	1,062	1,773
0.7	0.15	30	16	11	2,25	0,1	2,3	1,085	1,897
0.7	0.15	30	31	6	3,25	0,1	0	1,072	1,852
0.7	0.15	30	27	8	2,75	0,1	2,1	1,072	1,841
0.7	0.15	30	53	43	2,75	0,3	1,4	1,069	1,784
0.7	0.15	30	47	51	2,5	0,2	1,6	1,055	1,812
0.15	0.7	30	48	26	3,25	0,7	0,3	2,1	1,794
0.15	0.7	30	38	12	3,25	0,4	0,2	2,071	1,745
0.15	0.7	30	53	38	2,25	0,6	1,9	2,109	1,758
0.15	0.7	30	51	31	2,75	0,6	1,3	2,086	1,764
0.15	0.7	30	56	61	2	0,7	1,7	2,118	1,793
0.15	0.7	30	56	63	4	0,6	0	2,124	1,759
0.15	0.7	30	26	6	3	0,5	0,9	2,071	1,79
0.15	0.7	30	46	58	4	0,5	0,2	2,107	1,778
0.15	0.7	30	51	35	2,25	0,6	1,977	2,082	1,803
0.15	0.7	30	51	59	3,25	0,7	0,6	2,107	1,794
0.15	0.15	30	53	49	2	0,52	2	1,259	1,775
0.15	0.15	30	56	56	3,75	0,5	0,2	1,257	1,749
0.15	0.15	30	29	6	3	0,6	0,7	1,253	1,805
0.15	0.15	30	46	58	3,75	0,7	0,2	1,259	1,79
0.15	0.15	30	41	51	4	0,5	0,159	1,267	1,784
0.15	0.15	30	55	51	3,25	0,5	0,655	1,235	1,757
0.15	0.15	30	36	36	1,5	0,6	2,4	1,267	1,797
0.15	0.15	30	53	26	3,5	0,4	0	1,269	1,726
0.15	0.15	30	45	26	3,25	0,5	0,5	1,246	1,749
0.15	0.15	30	61	16	2,5	0,5	2,8	1,24	1,772