UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

GRADO EN INGENIERÍA DE COMPUTADORES

Análisis e implementación de un mecanismo de toma de decisiones para sistemas robotizados en entornos industriales

Analysis and implementation of a decision-making mechanism for robotic systems in industrial environments

Realizado por

Elizaveta Rivas Babintseva

Tutorizado por

Ana María Cruz Martín y Juan Antonio Fernández Madrigal

Departamento

Ingeniería de Sistemas y Automática

UNIVERSIDAD DE MÁLAGA

MÁLAGA, SEPTIEMBRE DE 2023

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADO EN INGENIERÍA DE COMPUTADORES

Análisis e implementación de un mecanismo de toma de decisiones para sistemas robotizados en entornos industriales

Analysis and implementation of a decision-making mechanism for robotic systems in industrial environments

Realizado por
Elizaveta Rivas Babintseva

Tutorizado por
Ana María Cruz Martín y Juan Antonio Fernández Madrigal

Departamento
Ingeniería de Sistemas y Automática

UNIVERSIDAD DE MÁLAGA
MÁLAGA, SEPTIEMBRE DE 2023

Resumen

El paradigma conocido como "Industria 4.0" se basa en la automatización y el uso de los datos en las tecnologías de producción [3]. Los fabricantes están adoptando este enfoque porque aporta ventajas como mayor automatización, mantenimiento preventivo, optimización automática de las mejoras de procesos, etc.; es decir, se aprovechan los recursos de la forma más inteligente posible. En este Trabajo Fin de Grado se ha estudiado una forma de integrar inteligencia artificial en este paradigma. Para ello, se ha implementado en el software *Visual Components* (VC) una celda de fabricación simulada para una línea de producción automática que puede tener varias estaciones de trabajo (almacenaje, transporte, mecanizado, ensamblaje e inspección) que se coordinan y conectan entre ellas, así como personal humano que trabaja junto con robots móviles. Este entorno simulado permite evaluar y optimizar la productividad, la calidad, los costos y la gestión de los productos cuando se usa un método de toma de decisiones óptimo bajo incertidumbre frente a la planificación heurística que haría el personal humano. También se ha programado en VC un mecanismo para la recogida de datos sobre la toma de decisiones del robot, que permite exportar los datos generados a herramientas externas para su análisis mediante un método de aprendizaje por refuerzo basado en modelo. Los resultados obtenidos con esta integración de inteligencia artificial en un entorno industrial demuestran que ésta mejora la toma de decisiones en la celda, optimiza el proceso de producción y usa más eficazmente los recursos.

Palabras clave: *Visual Components*, *MATLAB*, robot, celda de fabricación, aprendizaje por refuerzo, industria 4.0, toma de decisiones.

Abstract

The paradigm known as "Industry 4.0" is based on automation and the use of data in production technologies [3]. Manufacturers are adopting this approach because it brings advantages such as improved automation, preventive maintenance, automatic optimization of process, etc; in other words, resources are used in the most intelligent way. In this Final Degree Project we have studied a way to integrate artificial intelligence in this paradigm. For this purpose, a simulated manufacturing cell has been implemented in the Visual Components (VC) software for an automatic production line that can have several workstations (warehouse, transport, machining, assembly and inspection) that are coordinated and connected between them, as well as human staff working together with mobile robots. This simulated environment allows to evaluate and optimize productivity, quality, costs and product management when using an optimal decision making method under uncertainty versus the heuristic planning that would be done by human staff. A mechanism for collecting data on the robot's decision making has also been programmed in VC, which allows the data generated to be exported to external tools for analysis using a model-based reinforcement learning method. The results obtained with this integration of artificial intelligence in an industrial environment show that it improves decision making in the cell, optimizes the production process and uses resources more efficiently.

Keywords: Visual Components, MATLAB, robot, manufacturing cell, reinforcement learning, industry 4.0, decision making.

Índice

Capítulo 1

Introducción

En esta sección se explica por qué se ha elegido el tema del Trabajo Fin de Grado y cómo se relaciona con la Industria 4.0 y el aprendizaje por refuerzo. También se presentan los objetivos específicos que se pretenden alcanzar con el TFG y cómo estos objetivos contribuyen al avance del conocimiento en el área de estudio. Por último, se proporciona una visión general de la estructura del resto de capítulos de la memoria, indicando qué información se puede encontrar en cada uno de ellos y cómo están organizados.

1.1 Motivación

El paradigma conocido como "Industria 4.0" se basa en una mayor automatización de los procesos y en el empleo de máquinas y sistemas inteligentes. Para ello, las fábricas inteligentes pueden recopilar y combinar datos de producción con otros datos operativos de la empresa con el fin de mejorar la toma de decisiones, transformando así la forma en que las empresas producen, mejoran y distribuyen sus productos.

En este paradigma los fabricantes adoptan nuevas tecnologías como el internet de las cosas (IoT), la computación en la nube, métodos de inteligencia artificial y de aprendizaje automático, etc., en sus plantas de producción y operaciones. Una característica de las fábricas inteligentes es el análisis de datos para mejorar la

toma de decisiones. Los sensores integrados y las máquinas conectadas generan una gran cantidad de datos relevantes que pueden ser analizados para ayudar a los desarrolladores a identificar patrones y mejoras [1][2].

Los datos en tiempo real recopilados por los sensores, dispositivos y máquinas de la fábrica pueden ser utilizados directamente por otros activos de la planta o integrados con el software empresarial [3].

En la actualidad, se está observando un creciente uso de réplicas de software de entornos de fabricación, conocidas como "*digital twins*". Estas herramientas tecnológicas se destacan por su habilidad para analizar, simular y optimizar procesos de una manera más eficaz y realista. Al ofrecer una visión detallada de los procesos de fabricación, los gemelos digitales están ayudando a las empresas a mejorar su eficiencia, disminuir costos y aumentar la calidad del producto final [4].

1.2 Objetivos

El objetivo principal de este Trabajo Fin de Grado es implementar una celda de fabricación simulada con robots móviles que interactúen con el entorno, compuesto de una estantería, cajas, una entrada, una salida y un humano que retira las cajas en mal estado, con el fin de estudiar la integración de métodos de inteligencia artificial (*machine learning*, más concretamente aprendizaje por refuerzo) y cómo éstos permiten optimizar el proceso de producción.

Para llevar a cabo este estudio, en primer lugar, se ha analizado la viabilidad del software *Visual Components* [7] para su integración con el algoritmo destinado a optimizar el proceso, que está implementado en un entorno del software externo

(*MATLAB* [5]). Se han realizado pruebas generales para verificar el comportamiento correcto del entorno y de su enlace con *MATLAB*, y pruebas unitarias para la importación y exportación de datos.

Los datos generados en la simulación se han analizado en *MATLAB* para estimar un modelo de la dinámica [30] con incertidumbre de ese entorno industrial y posteriormente para encontrar la política óptima que optimiza el uso inteligente de los recursos, para lo que hemos empleado el método *Value Iteration* [6].

La nueva política ha sido incorporada en la celda simulada para su análisis y comparación con otras políticas, una implantada por una persona y otra puramente aleatoria.

Un resumen de los objetivos específicos es el siguiente:

1. Analizar la viabilidad del software *Visual Components* para desarrollar el algoritmo de inteligencia artificial destinado a la optimización.

2. Implantar un mecanismo de toma de decisiones del robot.

3. Analizar los resultados obtenidos utilizando *MATLAB* como una herramienta externa offline.

4. Encontrar la política óptima.

5. Implantar la política óptima en la celda simulada y volver a analizar los datos.

Es importante mencionar que la herramienta *Visual Components* aún no cuenta con este tipo de algoritmos integrados, de ahí el interés y utilidad de este estudio.

1.3 Estructura de la memoria

Este documento está estructurado en varias secciones que se presentan resumidamente a continuación.

En este primer capítulo se han expuesto la motivación y los objetivos que han guiado la realización del Trabajo Fin de Grado. A continuación, en el segundo capítulo, se detallan las tecnologías que se han utilizado en el TFG, explicando las razones de su elección y cómo han contribuido al desarrollo del mismo. En el tercer capítulo se describen el proceso de diseño y la implementación del software, proporcionando información detallada sobre cómo se ha llevado a cabo. El cuarto capítulo está dedicado a la creación y descripción de las pruebas, y el análisis de los resultados obtenidos. En el quinto capítulo se resumen las conclusiones obtenidas y cómo estas se relacionan con los objetivos iniciales planteados, así como posibles trabajos futuros que podrían llevarse a cabo para ampliar el presentado aquí.

Además, se incluye un apéndice con un manual de uso tanto de la celda de fabricación desarrollada en *Visual Components* como del proceso de análisis en *MATLAB,* para facilitar su utilización por parte de otros usuarios.

Capítulo 2

Herramientas

En este Trabajo Fin de Grado se pretende estudiar la posibilidad de usar un método de inteligencia artificial en la simulación de un entorno industrial para permitir la optimización del segundo. Para ello, se necesita integrar las dos herramientas principales consideradas, *Visual Components* y *MATLAB*. En este capítulo se presentan ambas herramientas y cómo funcionan a nivel general para este TFG. También se exponen otras posibles alternativas y se justifica la elección de *Visual Components* y *MATLAB* como las herramientas más adecuadas para este proyecto.

2.1 *Visual Components*

Visual Components es un software profesional industrial de simulación de fabricación. Ofrece capacidades avanzadas de simulación 3D, un motor de simulación física realista y opciones de programación flexibles para incluir diversos elementos presentados en entornos industriales. Su principal objetivo es simular líneas de producción o fábricas completas, incluyendo robots, vehículos autoguiados, cintas transportadoras y otros elementos, y proporcionar datos en tiempo real para su análisis [7]. Entre sus características más destacadas se encuentran:

- *ECatalogue*. Es una librería de modelos virtuales con más de 2500 componentes, incluyendo robots, *AGVs*, cintas transportadoras y máquinas

de las principales marcas de automatización como *ABB, KUKA, Toshiba* y *Schneider Electric*. La librería se actualiza regularmente y se puede acceder fácilmente desde la ventana principal de la aplicación. Los componentes están organizados por tipo o fabricante y se pueden buscar directamente mediante la barra de búsqueda. Además, el usuario puede personalizar los comportamientos y propiedades de los componentes del catálogo para crear una biblioteca personalizada.

- Importación y exportación de archivos. Permite la importación y exportación de archivos CAD [28] con el diseño del TFG realizado en múltiples formatos, incluyendo aquellos utilizados por los principales programas de diseño como *AutoCAD*, *Blender* y *CATIA* [29].

- Estadísticos. Permite la creación y exportación de gráficos y datos estadísticos obtenidos de la simulación en tiempo real para su análisis posterior.

- Plataforma abierta. La plataforma cuenta con una API de Python [7] que permite personalizar la aplicación y programar robots o acciones dentro del proceso de simulación, así como generar indicadores clave de desempeño adicionales. También cuenta con una API de .NET [8] que permite crear complementos, enviar mensajes y editar componentes creados durante la simulación desde un editor de código externo, como *Visual Studio Code* [9].

- Ayuda y tutoriales. La plataforma ofrece una amplia variedad de recursos de ayuda y tutoriales, tanto dentro del programa como en su sitio web. Los usuarios pueden acceder a cursos gratuitos, manuales y documentación para aprender sobre el funcionamiento del programa a nivel básico y avanzado.

También hay recursos disponibles en plataformas gratuitas como YouTube y en foros de discusión.

En la Tabla 2.1 se muestra una comparativa entre programas de simulación con características similares. Para este TFG, se ha escogido *Visual Components* por su API abierta, que facilita la integración con los métodos externos de inteligencia artificial, y porque la Universidad de Málaga ofrece licencias en red gratuitas para acceder a una versión completa del programa.

Características	***Visual Components*** **[10]**	***Simul8*** **[11]**	***Flexsim*** **[12]**
Librerías propias	Sí	Sí	Sí
Importación/Exportación de archivos	Sí	Sí	Sí
Exportación estadísticos	Sí	Sí	Sí
Datos en tiempo real	Sí	Sí	Sí
Plataforma abierta	Sí, Python	No	Sí, C++
Ayuda y tutoriales	Sí	Sí	Sí
Simulación 3D	Sí	Sí	Sí
Precio Inicial	Estudiante básico gratuito. Estudiante UMA premium gratuito.	4995,00 US$/año	-
Versión	Prueba Gratuita	Prueba Gratuita y Versión Gratuita	Prueba Gratuita y Versión Gratuita
Soporte Software	*Windows*	*Windows, Online Web, MacOS*	*Windows*
Idioma (Interfaz de usuario)	Español, Inglés	Español, Inglés	Español, Inglés

Tabla 2.1 Comparación de características entre los programas de simulación evaluados.

Es importante mencionar que se ha utilizado la versión 4.7 de *Visual Components*, disponible en `software.uma.es` a través del Servicio Central de Informática de la UMA [20]. Al inicio del Trabajo Fin de Grado se trabajó con la versión 4.3, pero debido a varios errores que afectaban el funcionamiento del TFG, como el mal tratamiento de las colisiones entre el humano y el robot, se optó por una versión más reciente que soluciona estos problemas específicos y garantiza el objetivo principal que se quiere lograr.

En el Apéndice A, *Manual de utilización de la celda de fabricación y el entorno de análisis*, se detallan los pasos y los requerimientos para trabajar con *Visual Components* usando la cuenta de la Universidad de Málaga.

2.1.1 Terminología

El Software *Visual Components* tiene una gran complejidad, lo que permite su flexibilidad y potencia. Para comprender su funcionamiento, es importante explicar los conceptos básicos que maneja [7]:

- *Componente.* Una representación de un objeto tridimensional. Ejemplo: un robot.
- *Comportamiento.* Acción o tarea que puede realizar el componente. Ejemplo: un robot desplazándose de un punto A hacia un punto B.
- *Controlador.* Un componente que puede ser utilizado para controlar el comportamiento de otro componente durante una simulación. Ejemplo: un controlador de un robot.
- *Espacio.* Un área rectangular ajustable (componente) en la que los otros componentes pueden moverse.

- *Señal.* Evento que se puede utilizar para controlar la ejecución del código interno del programa. Ejemplo: una señal que indique cuándo la batería está baja. De esta manera, el robot podrá dirigirse a la estación de carga para recargar su batería y evitar cualquier otro comportamiento.
- *Ejecutor.* Una clase que permite ejecutar un programa del componente.
- *Proceso.* Definición abstracta de una acción o conjunto de acciones que puede llevar a cabo una máquina. El término «proceso» puede referirse a la implementación del proceso o al grupo de procesos, dependiendo del contexto. Ejemplo: un proceso llamado estantería que, al llegarle un componente, lo guarde.
- *Editor de procesos.* Interfaz gráfico de usuario que permite diseñar y modificar procesos (véase Figura 3.3).
- *Declaración (de proceso).* Un solo paso de ejecución en un proceso.
- *Rutinas (de proceso).* Lista de declaraciones guardadas en el ejecutor.
- *Implementación del proceso.* Una rutina concreta del proceso en un ejecutor.
- *Variables.* Valores que se pueden utilizar en la programación de robots y en la configuración de procesos para almacenar y manipular información durante la simulación.
- *Producto.* Por lo general, se refiere a un ejemplo del producto, pero también se puede referir al tipo de producto en ciertos contextos.
- *Flujo.* Secuencia de transferencia de un componente o componentes lógicamente hacia y desde otros comportamientos (véase Figura 3.2).
- *Grupo de flujo del proceso.* Una agrupación de tipos de productos utilizados para definir las secuencias de flujo del proceso y el flujo del material. Cada tipo de producto es miembro de un grupo de flujo concreto en todo momento.
- *Secuencia de flujo del proceso.* Una secuencia de pasos de flujo definida para un grupo de flujo del proceso en el editor de flujo del proceso.

2.1.2 Interfaz de usuario de *Visual Components*

En este apartado, se describen las diferentes pestañas disponibles en *Visual Components* [7].

- Pestaña Inicio (véase Figura 2.1). En esta sección, se pueden establecer interfaces entre el controlador y el robot. Es importante destacar que en este TFG se utilizan 3 controladores: 2 para los robots y 1 para el humano. Estos controladores permiten la interconexión del robot con sus puntos de recarga, el punto de reposo y el espacio que puede seguir a través de interfaces (consulte la Figura 2.1). Si no se ha definido el espacio disponible mediante el componente *Pathway Area*, los componentes vinculados no podrán moverse. Para realizar la vinculación, es necesario acceder a la sección *Conectar* dentro de la pestaña de inicio, ubicada en la parte superior derecha, y seleccionar la opción de interfaces (véase Figura 2.2). Al hacer pinchar en ella, se debe seleccionar uno de los controladores y se mostrará una interfaz de usuario sencilla e intuitiva para vincular los objetos controlables con los que están presentes en el TFG (véase Figura 2.1.)

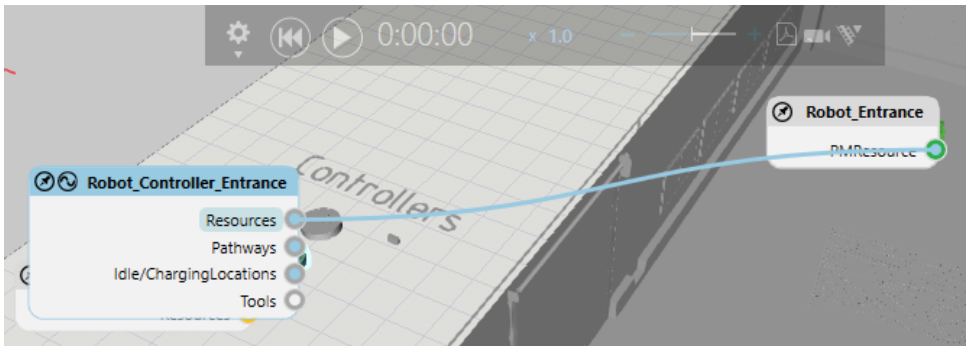


Figura 2.1 Interfaz de relación del controlador del robot.

En la misma pestaña, también es posible gestionar la importación y exportación de estadísticas. Para ello, se debe acceder a la sección

Estadísticos, ubicada en la parte superior derecha, donde se encuentran opciones para importar/exportar datos, seleccionar el intervalo de tiempo para la toma de datos y visualizar las estadísticas en tiempo real. Es importante tener en cuenta que solo se exportan los datos recopilados hasta el momento (véase el apartado de *Visual Components* en el Apéndice A, *Manual de utilización de la celda de fabricación y el entorno de análisis*).

Figura 2.2 Interfaz de la pestaña de inicio.

- Pestaña Proceso (véase Figura 2.3). En esta pestaña, se especifica el flujo de trabajo y los productos definidos. Para visualizar los productos y flujos de trabajo existentes, es necesario acceder a la sección *Productos*, ubicada en la parte superior derecha, donde se muestran tanto el flujo de trabajo como los objetos utilizados en su interior (véase Figura 2.3).

  Para ver los procesos usados en este trabajo, se debe utilizar la sección llamada *Procesos*, ubicada junto a la sección *Productos* (véase Figura 2.3). Al seleccionar uno de los procesos, se mostrarán tanto las variables definidas temporalmente en él como el editor de procesos.

  Finalmente, junto a la sección de *Procesos* se encuentra la sección llamada *Flujo* para editar todo lo que tenga que ver con esto. Existen varias formas de trabajar con esta herramienta que se detallarán en el Apéndice A, *Manual de utilización de la celda de fabricación y el entorno de análisis*.

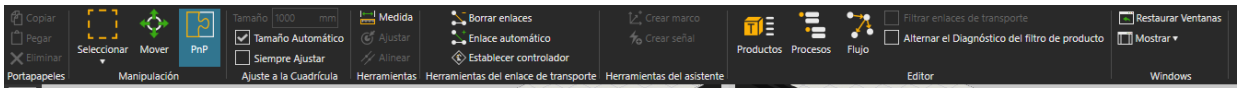


Figura 2.3 Interfaz de pestaña de procesos.

- Pestaña Modelado (véase Figura 2.4). Esta pestaña está diseñada para permitir la edición del código de los componentes utilizados, así como para gestionar las señales y las variables empleadas. Para visualizar la implementación de la toma de decisiones y el almacenamiento de datos, es necesario seleccionar el componente. Al hacerlo, aparece una pestaña en la parte izquierda denominada *Gráfico de componentes*, que muestra las propiedades y el comportamiento definido para este objeto:
    - Las propiedades son variables que pueden ser definidas por el usuario o establecidas por defecto para garantizar el correcto funcionamiento de un objeto. Estos valores son fijos y no pueden ser modificados durante la ejecución del programa.
    - El comportamiento incluye tanto las señales definidas como el código escrito en Python para gestionar el funcionamiento interno del componente.

  Dentro del código del componente se encuentra el encargado de gestionar el almacenamiento y la importación de datos, así como los valores de las señales que pueden ser recibidas y enviadas. Se ubica en el archivo denominado *ResourceScript*.

  Para añadir cualquier tipo de comportamiento o propiedad, es posible hacerlo a través de la ventana correspondiente ubicada en la parte superior derecha, seleccionando el tipo deseado, como una señal con un valor numérico real o un nuevo código.

Figura 2.4 Interfaz de pestaña de modelado.

- Pestaña Ayuda (véase Figura 2.5). Se proporciona una guía de Python con la versión específica utilizada, así como información sobre el funcionamiento del programa. Además, se ofrece material de soporte en línea, como un foro, una academia con grabaciones y ejemplos prácticos, y blogs. También se comparten contenidos en redes sociales como *YouTube, Facebook, LinkedIn y X (Twitter).*

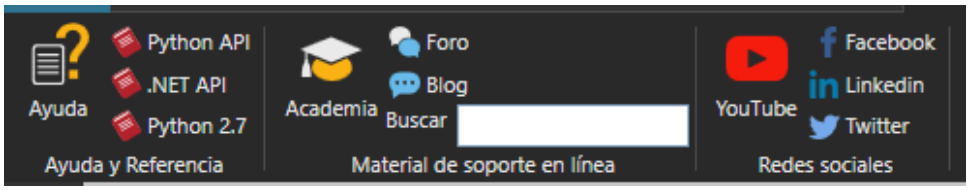


Figura 2.5 Interfaz de pestaña de ayuda.

2.2 *MATLAB*

MATLAB es un sistema de cómputo numérico que proporciona un entorno de desarrollo integrado con lenguaje de programación propio. Está disponible en múltiples plataformas, incluyendo *Unix, Windows, MacOS y GNU/Linux.* Sus capacidades incluyen la manipulación de matrices, visualización de datos y funciones, implementación de algoritmos, creación de interfaces de usuario y comunicación con otros programas y dispositivos hardware [5].

Aunque existen alternativas gratuitas y de código abierto como *GNU Octave y R* (véase Tabla 2.2), *MATLAB* es una herramienta poderosa para la creación de gráficos y el análisis estadístico, de ahí el interés de esta herramienta ya que permite la exploración de patrones y tendencias en los datos a través de una amplia gama de funciones y herramientas integradas. Con *MATLAB*, puede personalizar los gráficos con nombres de casos, líneas de mínimos cuadrados y curvas de referencia, y calcular estadísticas descriptivas como el valor máximo, promedio, mediana, mínimo, modo, desviación estándar y varianza.

El propósito principal de *MATLAB* en este TFG es analizar un modelo de la dinámica [30] con incertidumbre en un entorno industrial, encontrar la política óptima que optimice el uso inteligente de los recursos y compararla con otras nuevas políticas mediante la creación de gráficos y análisis estadístico.

Características	***MATLAB*** **[13]**	**GNU Octave** **[14]**	***R*** **[15]**
Licencia	Comercial	Libre y gratuita	Libre y gratuita
Lenguaje de programación	Lenguaje M	Lenguaje M (compatible con *MATLAB*)	*R*
Manipulación de matrices	Sí	Sí	Sí (con la librería *matrix*)
Representación de datos y funciones	Sí	Sí	Sí
Implementación de algoritmos	Sí	Sí	Sí
Creación de interfaces de usuario	Sí (con GUIDE)	Sí (con *QtOctave*)	Sí (con librerías como *shiny*)

Precio	Comercial, con diferentes opciones de licencia y precios. UMA estudiante gratuito	Libre y gratuito	Libre y gratuito
Soporte Software	*Windows, Online Web, MacOS y Linux (algunas distribuciones)*	*Windows, MacOS y Linux*	*Windows, MacOS, Linux y Unix*
Idioma (interfaz de usuario)	Español, Inglés	Español, Inglés	Español, Inglés

Tabla 2.2 Comparación de características entre los programas de análisis matemático evaluados [16].

En el Apéndice A, *Manual de utilización de la celda de fabricación y el entorno de análisis*, se detallan los pasos y los requerimientos para trabajar con la herramienta usando la cuenta de la Universidad de Málaga. Para este trabajo se ha usado la versión online de *MATLAB* R2023a, por sus ventajas como el uso de los archivos generados desde cualquier lugar, sin necesidad de descargar el software y más rápida que la versión de escritorio. También se ha usado debido a problemas técnicos usando la versión escritorio con el ordenador usado para la realización de este trabajo.

2.2.1 Elementos básicos del entorno

A continuación, se describen los diferentes elementos de *MATLAB*.

- Escritorio (véase Figura 2.6). Hay varias partes a destacar [23]:
    - *Command Window* (color marrón). La sección donde se ejecutan todas las instrucciones y programas. Puede usarse para cálculos rápidos. También puede mostrar los últimos comandos ejecutados para comprobar las acciones recientes y re-ejecutar instrucciones.

- *Workspace* (color morado). Muestra las variables que se están usando, su tipo, sus dimensiones (si son matrices) y su valor actual.
- *Current directory* (color verde).Ventana de navegador que muestra la ruta actual.
- *Files* (color azul). Muestra y administra todos los archivos y carpetas que se encuentran subidos en el TFG. No siempre está ubicado en el *current directory.*
- *Archivos abiertos* (color naranja). Ventana que muestra los archivos abiertos ya sea para su visualización o edición.

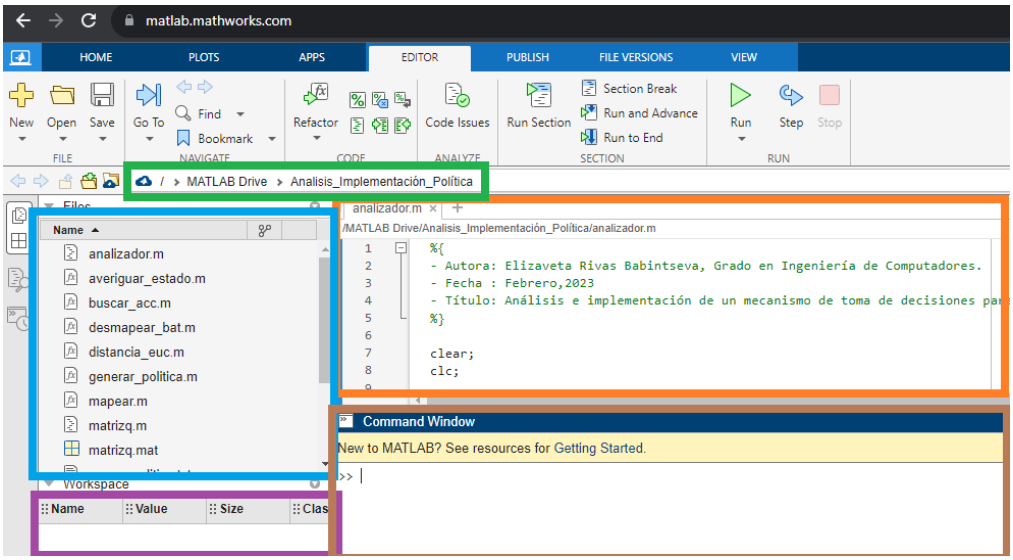


Figura 2.6 Interfaz de usuario del escritorio de *MATLAB*.

- Importación y exportación de datos. *MATLAB* proporciona varias funciones para importar y exportar datos.

  La importación de datos en el área de trabajo de *MATLAB* se puede hacer de varias maneras, dependiendo del archivo y del formato de este a través de funciones como `importdata(), readtable(), load(), etc.`

  Para la exportación de datos, se puede crear un código que recorra los datos y los guarde usando *fprint()*(u otro método) en un archivo con la extensión

deseada, teniendo en cuenta que, si se utiliza la versión online de *MATLAB*, se guardarán en *MATLAB* Drive [24] y no en el equipo local.

- Generación de gráficas de los datos. Después de importar los datos en el área de trabajo de *MATLAB*, se pueden representar los datos para explorar sus características. Para ello, se puede utilizar la función `plot,` la cual crea gráficas de líneas bidimensionales, con personalización de colores, líneas, marcadores y su facilidad de usarlo [25].
- Vectores. Para la creación de los vectores se encierran los elementos entre corchetes y se usa espacio o coma para vectores fila (ej., r = [7 8 9 10 11]) y punto y coma para vectores columna. Para el acceso a estos se usan los paréntesis y dentro de estos, el índice del elemento del vector a referir (ej., r = [7 8 9 10 11] ; r(2)= 8) [26].

- Matrices. Crear una matriz en *MATLAB* es tan fácil como crear un vector, con puntos y comas (;) para separar las filas de una matriz. Para representar matrices tridimensionales el procedimiento es algo más complejo ya que no hay una función específica y, por tanto, requiere hacer 3 bucles recorriendo tanto las filas, las columnas y la tercera dimensión [27].

Capítulo 3

Diseño e implementación

En este TFG se pretende estudiar la posibilidad de usar un método de inteligencia artificial en la simulación de un entorno industrial para permitir su optimización. En este contexto, el objetivo es generar una política óptima para la línea de producción. Esto requiere integrar las dos herramientas principales consideradas, *Visual Components* y *MATLAB*. En este capítulo se describe en detalle el trabajo realizado en cada una de ellas para llevar a cabo dicha integración.

3.1 Simulación del entorno industrial en *Visual Components*

El objetivo de utilizar *Visual Components* es simular una celda de fabricación, programar un mecanismo para recopilar datos sobre la toma de decisiones del robot, exportar los datos generados para encontrar la política óptima. Una vez encontrada, se inserta la nueva política en *Visual Components* para volver a analizar los datos.

Para la creación de la celda de fabricación se ha utilizado una representación digital de un almacén real. Se han incluido dos robots móviles, junto con sus respectivos puntos de recarga y descanso, una estantería, una entrada de productos, una salida y un operador humano (véase Figura 3.1):

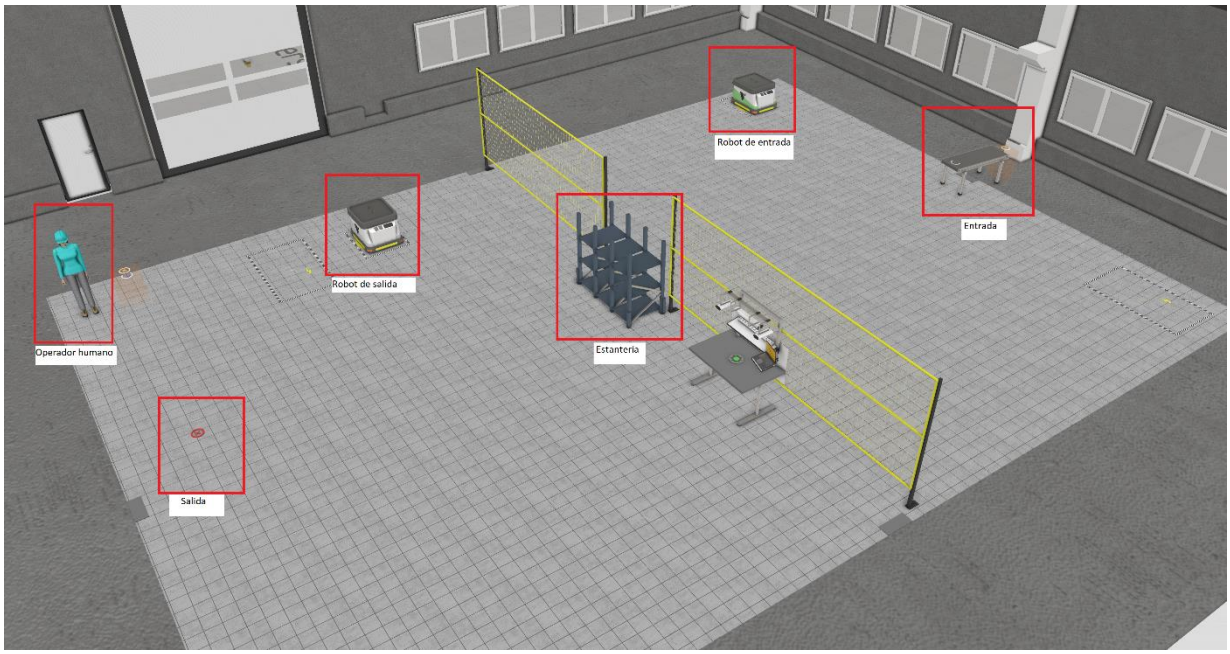


Figura 3.1 Celda de fabricación.

- *Caja.* Es el producto manejado en todo el almacén, tanto por robots como por humanos, y se almacena en las estanterías.
- *La entrada de productos.* Está programada para generar cajas en intervalos y cantidades aleatorias. Esta fuente de productos está conectada a una cinta transportadora, lo que permite que el robot de entrada pueda cargar las cajas generadas desde dicha cinta.
- *El robot de entrada.* Está programado para recoger cajas de la cinta trasportadora, con una capacidad máxima de dos unidades, y transportarlas a la estantería. Si la estantería está llena, el robot no debería recoger más cajas, sino esperar en su punto de descanso hasta que haya al menos un espacio disponible para depositar las que lleguen desde la entrada. De igual manera, si no hay cajas disponibles para ser recogidas, el robot debería permanecer en reposo.

- *La estantería.* Sirve como almacenamiento temporal para las cajas. Está dispuesta para evitar interferencias entre los robots de entrada y salida al manipular las cajas. Este componente cuenta con señales internas (véase Tabla 3.2), tanto creadas por la autora como ya predefinidas, para enviar y recibir información relevante para el análisis de los datos.
- *El robot de salida.* Está programado para tomar varias decisiones, como recoger una o dos cajas, dirigirse al punto de descanso o recargar (véase Tabla 3.1). La aplicación de todas las decisiones se detallará en la sección de funcionamiento. Este componente cuenta con señales para enviar y recibir información relacionada con la supervisión de la batería, la interacción con humanos y el análisis estadístico.
- *Salida.* Este componente recoge las cajas depositadas por el robot de salida y simula su salida definitiva del almacén. Cuenta con un sensor que envía y recibe información sobre el número total de cajas que han salido del almacén.
- *Operador humano.* Existe un humano que se desplaza continua y aleatoriamente entre dos puntos de la zona donde opera el robot de salida. Esto se hace para introducir incertidumbre en la navegación de los robots. En ningún momento el humano recoge cajas de la estantería ni interfiere con las cajas manipuladas por los robots. Las cajas que en teoría son utilizadas por el humano son distintas de las que manipulan los robots y se generan fuera de los elementos mencionados anteriormente.

Acción	Descripción
1	Coger 1 caja
2	Coger 2 cajas
3	Descansar 50 segundos
4	Recargar batería

Tabla 3.1 Decisiones disponibles en el robot de salida.

Un punto importante a destacar es que, en varios experimentos, se ha demostrado que el tiempo mínimo necesario para pasar de un estado a otro es de 50 segundos. Por esta razón, se eligió este tiempo para la elección 3, como se puede ver en la Tabla 3.1.

En relación al tema de la batería y la recarga de un robot, es importante mencionar que todos los robots comienzan con la capacidad máxima de batería y cuentan con una base propia para recargarla y descansar. Tras completar una acción, ya sea almacenar una caja en la estantería o llevarla a la salida, los robots hacen uso de estas bases. Si el nivel de batería del robot es inferior al 20% y está realizando una acción, está programado para completarla antes de dirigirse a la base de recarga. Al llegar a la base, es probable que el nivel de batería sea del 10% o menos, pero nunca sin que se quede sin batería por completo siempre y cuando no se cambie el dimensionamiento del espacio creado en la celda de fabricación. Bajo ningún concepto puede dejar una acción a medias. Al recargar, ya sea por haber completado una acción o por tener un nivel bajo de batería, el robot espera un lapso de 120 segundos y, tras evaluar si el nivel sigue siendo bajo, continúa recargando hasta alcanzar el 100% de su capacidad.

Cabe aclarar este comportamiento ocurre siempre incluso cuando obedece una política que se le da desde fuera.

En cuanto al trabajo con *Visual Components*, se descubrió inicialmente que la batería del robot podía llegar a ser negativa, lo cual era inadecuado. Por esta razón, se programó el robot para que nunca pudiera quedarse sin batería en medio de una acción. Esta regla está preparada específicamente para este entorno.

El funcionamiento deseado del almacén se describe a continuación:

1. El robot de entrada recoge los paquetes y los transporta a un estante para su almacenamiento. En caso de necesidad de recarga, el robot va a su punto de recarga cuando su batería está por debajo del 20% , aunque haya cajas en espera.
2. El robot de entrada permanece en espera hasta la llegada de los paquetes en la entrada del almacén.
3. Tras un período de tiempo determinado, el robot de salida tomará una decisión basada en la política que se esté siguiendo. Esta decisión puede incluir recoger una o dos cajas y transportarlas hacia la salida, descansar 50 segundos o recargar la batería. Una vez tomada la decisión, el robot procederá a la siguiente acción, teniendo en cuenta el caso especial de batería baja mencionado anteriormente.
4. Tras la toma de decisión por parte del robot, se recopilan todos los datos necesarios para su análisis mediante señales o variables del entorno previamente establecidas.

En este Trabajo Fin de Grado se han declarado 5 procesos de *Visual Components* (véase apartado 2.1.1): las cajas, la entrada de productos y salida, la estantería y dos más para el funcionamiento del humano.

Para definir el flujo de trabajo, se dispone de una herramienta llamada *Flujo* (véase apartado 2.1.1), que permite construir varios flujos de manera gráfica utilizando los procesos mencionados anteriormente junto con el producto usado. En nuestro caso concreto, existen dos grupos de flujo de los procesos: el primero para el transporte de cajas por parte de los robots y el segundo para el flujo del humano. La Figura 3.2 muestra la secuencia de flujo del proceso y cómo se ve la transición del proceso de la entrada de productos hacia la salida en el TFG para el caso de los robots móviles.

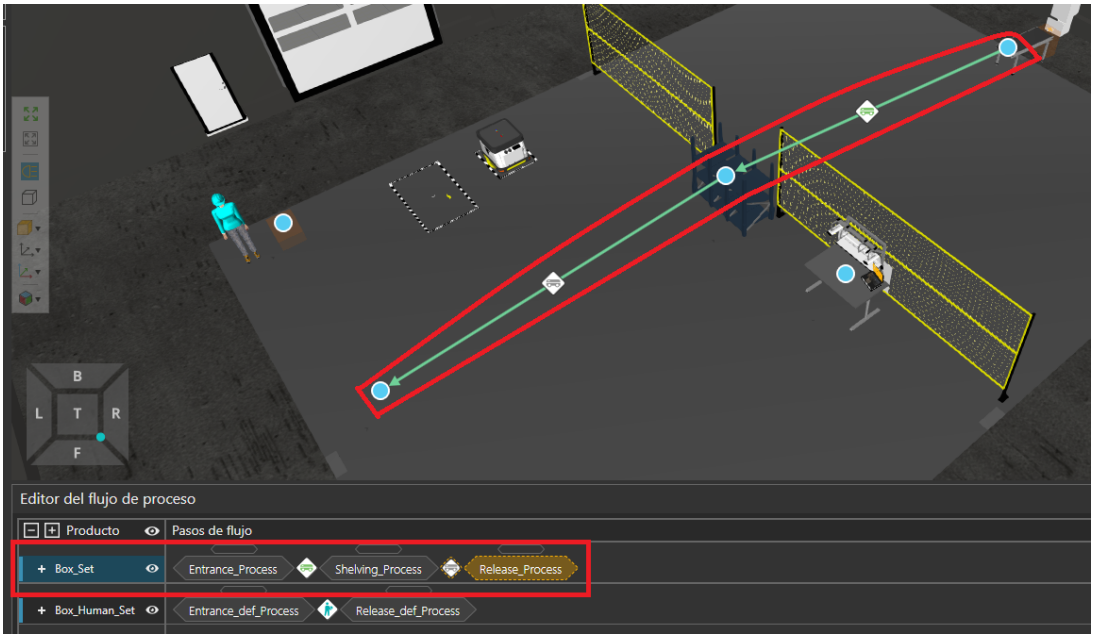


Figura 3.2 Interfaz de usuario del editor de flujo del proceso.

Para la toma de decisiones, *Visual Components* dispone de una herramienta denominada `ProcessExecutor_HIDE_`, que actúa como un editor de procesos y facilita la representación gráfica de la rutina del proceso.

En la Figura 3.3 se presenta la interfaz de usuario correspondiente a la implementación del proceso de salida, puesto que es en esta etapa donde se llevará a cabo la configuración, envío y recepción de señales (véase Tabla 3.2) del estado actual del robot de salida y de la estantería, así como la toma de decisiones, el estado del robot de salida y de la estantería después de la toma de decisión y otros elementos esenciales para el adecuado desempeño del TFG.

Ubicación : nombre Señal	**Descripción**
Shelving Box : Count Box bfr	Contador del número de cajas en la estantería antes de la toma de decisión.
Shelving Box : Count Box aft	Contador del número de cajas en la estantería después de la toma de decisión.
Robot Controller : Battery bfr	Nivel de batería discretizado del robot antes de la toma de decisión.
Robot Controller : Battery aft	Nivel de batería discretizado del robot después de la toma de decisión.
Robot Controller : Action	La decisión tomada (véase Tabla 3.1)

Tabla 3.2 Señales.

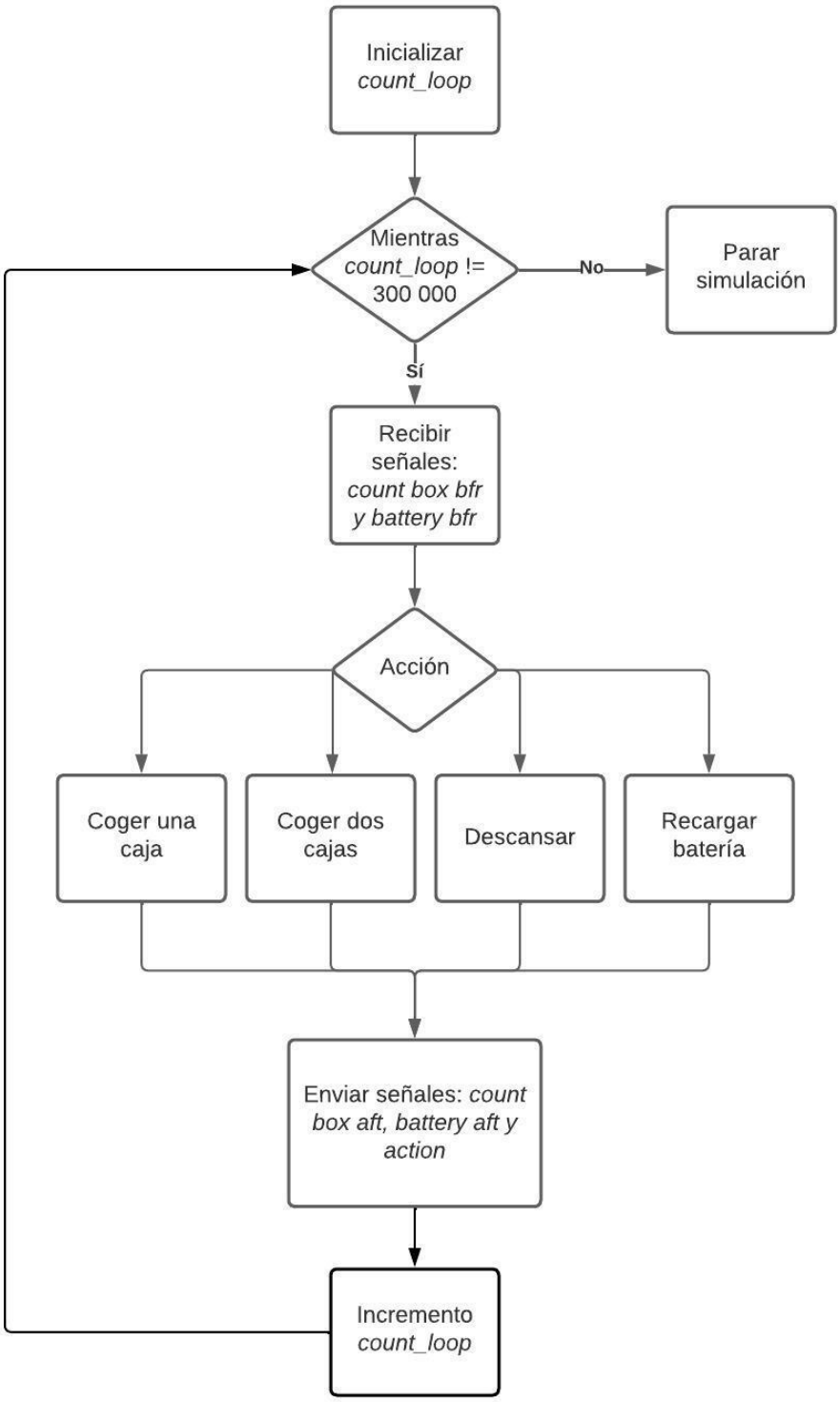


Figura 3.9 Diagrama de flujo del editor de proceso.

Figura 3.3 Editor del proceso de salida.

En la Figura 3.3 se muestran varias variables predefinidas en el componente. El propósito principal de estas variables es almacenar temporalmente los valores de las señales recibidas y enviarlas hacia afuera (véase Tabla 3.2). La idea detrás de esta lógica es la siguiente(véase Figura 3.9):

1. Existe una variable llamada `count_loop` que se utiliza para llevar un registro del número de tomas de decisiones realizadas por el robot de salida hasta el momento. En el capítulo 4, esta variable se denomina iteraciones de simulación.
2. Se registran las señales del nivel de batería en ese estado, el número de cajas en la estantería y, de acuerdo con la política establecida, el número que indica qué decisión se llevará a cabo. Existe un aspecto crucial en estas señales, que se derivan del nivel de batería en la toma de decisión 1 de la ejecución. A la hora de tomar y guardar la señal del nivel de batería antes de la toma de decisión, esta se inicia de manera adecuada debido a que no le ha dado tiempo a refrescarse. Por lo tanto, ha sido necesario corregir la señal.
3. Una vez determinada la toma de decisión a realizar, se lleva a cabo una de las acciones: recoger una o dos cajas y transportarlas hacia la salida, descansar 50 segundos o recargar la batería.
4. Luego, los datos generados dentro del programa interno del robot se recopilan en forma de matriz y se envían como señales a variables dentro del robot de salida para su posterior almacenamiento.
5. Se incrementa el valor del contador de tomas de decisiones y el ciclo se repite.

Para llevar los datos recopilados al exterior con el fin de que los utilice *MATLAB*, se ha implementado una función denominada "guardado". Esta función declara una variable llamada `Poli_num` dentro del código interno del robot de salida. Cuando *Visual Components* se usa para aprender un modelo de la dinámica del entorno, la política de acción se inicializa con un valor obtenido de una distribución de probabilidad uniforme discretizada entre 1 y 4. Esto permite realizar una primera toma de decisión en el editor `ProcessExecutor_HIDE_`.

Posteriormente, se almacenan en una matriz los resultados de la acción tomada, el nivel de batería antes y después de la toma de decisión, y el número de cajas en la estantería del almacén antes y después de realizar la acción. Estos datos se guardan en un archivo Excel, disponible en *Visual Components*, que se transfiere a *MATLAB* para su procesamiento.
Cuando *MATLAB* genere una política conocida ésta se extraerá e integrará en el programa interno del robot permitiendo que tome decisiones en base a la dicha política.

3.2 Cálculo de la política óptima en *MATLAB*

Una vez recopilados todos los datos necesarios en *Visual Components*, se exportan para que el método de aprendizaje por refuerzo implantado en *MATLAB* deduzca un modelo de la dinámica del entorno [30] y encuentre una política óptima para dicho modelo. Cuando se haya generado esa política, se exportará a su vez a VC, y se modificará el método de toma de decisiones del robot para que la ejecute.

Los datos registrados por *Visual Components* se corresponden con los valores de las señales mencionadas en la Tabla 3.2:

- La señal del nivel de batería, tanto antes como después de elegir y llevar a cabo una acción. Dado que la batería tiene una capacidad de 100 y que el método de aprendizaje por refuerzo utilizado es tabular, es decir, requiere un conjunto discreto de valores para representar estados y acciones, sus valores se han discretizado, tal como se muestra en la Tabla 3.3.

Rango de discretización	**100-80**	**80-60**	**60-40**	**40-20**	**20-0**
Valores discretizados	5	4	3	2	1

Tabla 3.3 Discretización de la señal del nivel de batería.

- La señal del número de cajas en la estantería, tanto antes como después de elegir y llevar a cabo una acción. La estantería tiene una capacidad máxima de 9 cajas.
- La acción que se toma en ese momento, de entre las cuatro disponibles (véase Tabla 3.1).

Por tanto, para obtener la política óptima, el primer paso es crear una matriz tridimensional de visitas de los distintos estados del sistema, a partir de la cual se puede deducir la dinámica, es decir, la probabilidad de llegar a un estado si se parte de otro y se realiza una acción. Es importante mencionar el hecho de que, para la obtención de la política óptima, se hace una toma de decisiones aleatorias con el fin

de recabar información de la dinámica del entorno, tal y como se ha explicado en el apartado anterior dedicado a *Visual Components*.

Lo primero que tenemos que hacer es definir los estados del sistema. Éstos deben incluir información del nivel de batería actual y de la cantidad de espacio ocupado en la estantería. El código de la función de *MATLAB* que obtiene un índice único de estado para un nivel de batería y un número de cajas en la estantería dados se muestra en la Figura 3.4.

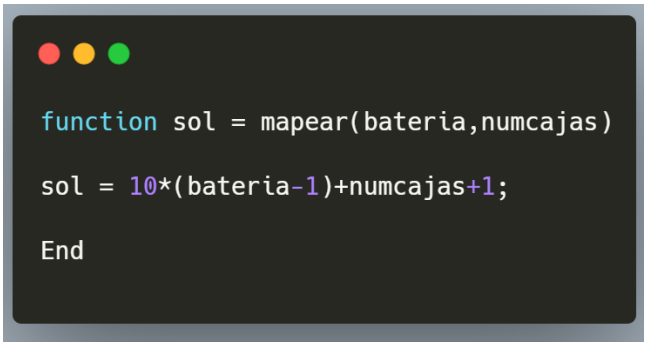


Figura 3.4 Función de mapeo del nivel de batería y cajas a un estado.

Esto da lugar a 50 estados posibles en la celda de fabricación, resultantes de la combinación de 5 estados de la batería y 10 estados de la estantería. También existe una función de desmapear que permite determinar el nivel de batería y el recuento de cajas en la estantería a partir del estado actual, como se puede ver en la Figura 3.8.

Tras ejecutar una serie de veces con las acciones aleatorias para obtener una muestra grande en *Visual Components*, se importan los datos obtenidos los cuales nos permiten formar una matriz S x A x S indexada por triplas (s0,a,s1), donde S0 es el estado actual, A la acción realizada y S1 el estado al que se ha llegado tras realizar esa acción. Cada celda contendrá el número de visitas que se han observado para dicha tripla (véase Figura 3.5).

```
function tabla = desmapeo(Nestados)

    for estado=1:Nestados
            % Nivel de bateria
            tabla(estado,1)=round(((estado-mod(6,9))/10)+1);
            % Número de cajas
            tabla(estado,2)=estado - 1 - (10 * (tabla(estado,1) -1));
    end
end
```

Figura 3.8 Función del desmapear de un estado.

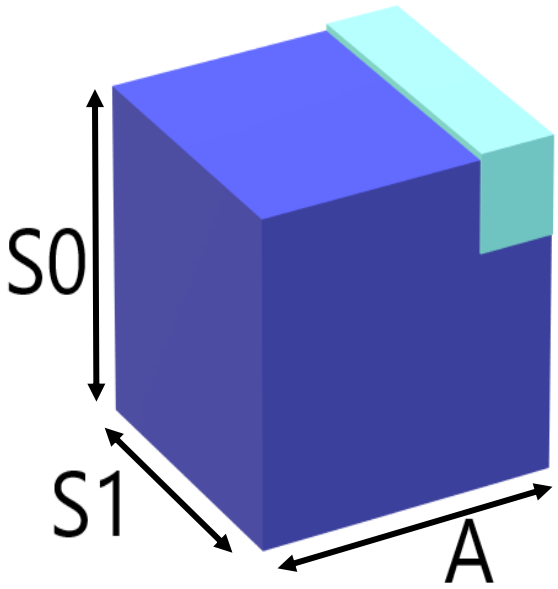


Figura 3.5 Matriz tridimensional de visitas.

Para obtener el modelo del sistema (de su dinámica), es necesario calcular la matriz tridimensional de probabilidad de transición P(S1 | S0, A). Para ello, se utiliza una matriz auxiliar bidimensional llamada matriz proyectada de visitas. Se selecciona la matriz de visitas original y se recorren todas las parejas (S0, A) para extraer el vector o "fibra" que contiene las visitas a otros estados generados por esa pareja.

Este vector se suma y el resultado se almacena en la matriz proyectada de visitas. Luego, este resultado se utiliza como divisor de las visitas de cada celda de la matriz original y obtener así las probabilidades buscadas.

Es importante destacar que cuanto mayor sea el número de visitas en la matriz bidimensional de visitas, más preciso será el resultado obtenido. Como mínimo, se recomienda que cada pareja tenga alrededor de 25 visitas, y un buen promedio sería de 50.

Una vez aprendido el modelo de la dinámica del sistema (la función P(S1 | S0, A)),utilizamos *Value Iteration* para encontrar la política óptima para dicho modelo. Es importante recordar que la política obtenida será un vector con tantas entradas como estados y que contendrá la acción decidida para cada uno.

El método utilizado de aprendizaje por refuerzo es un método basado en modelo que estima la dinámica del entorno estadísticamente y posteriormente busca una política óptima para dicha dinámica basándose en el algoritmo *Value Iteration*, que a su vez es una forma de programación dinámica [17]. Para ello hay que definir la recompensa y el factor de descuento.

Esta recompensa, R (S0, A, S1), es habitualmente un valor numérico real y se concede en función del estado inicial, la acción tomada y el estado final resultante, y se le dará al agente en cada paso. Lo habitual es basarla en la maximización de una recompensa esperada hasta un horizonte infinito, ponderando menos (descontando) la que pueda obtenerse más a largo plazo con el fin de poder encontrar una política óptima hay que definir esa optimalidad.

En nuestro caso, incluirá una penalización si se llena de la estantería de cajas y un valor neutro (0) en cualquier otra circunstancia (ver Tabla 3.4). El análisis de la Tabla 3.4 indica que la selección de la acción 4 (véase Tabla 3.1), correspondiente a la recarga, puede resultar desfavorable debido a la acumulación de cajas y su

consecuente inhabilitación. Sin embargo, esta acción podría tener un impacto positivo a largo plazo al permitir que el dispositivo esté completamente cargado, gracias al factor de descuento.

Recompensa	Condición del estado inicial S0
-10	Si está llena la estantería
0	Otro caso

Tabla 3.4 Recompensas.

El valor del parámetro gamma $\gamma \in (0,1)$, que representa el factor de descuento al que se someten las recompensas futuras, será de 0,9, que es un valor estándar. Cuanto más alto sea, más se tendrá en cuenta el futuro.

Una vez definida la recompensa y su descuento, el algoritmo *Value Iteration* busca estimar lo que se llama una función de valor-acción, que es una matriz Q de estados por acciones que guarda el valor estimado hasta este momento que tiene escoger la acción correspondiente a una celda dado que estemos en el estado correspondiente a la misma. Este valor se entiende como recompensa esperada acumulada hasta el infinito con el apropiado descuento. De forma que, en cada estado, se guarda en cada celda (S0, A) el valor que tiene la política actual, llamada $Value_{\pi}$, para esa celda, representado por:

$$Value_{\pi} = r_k + \gamma \cdot r_{k+1} + \gamma^2 \cdot r_{k+2} + \cdots + \gamma^{\infty} \cdot r_{\infty}$$

Para cada estado y acción en la matriz Q, se asigna un valor en la iteración k+1 de la celda (S, A) teniendo en cuenta todos los estados posibles de destino S1 de la matriz de probabilidad de transición junto con el valor anterior de esa celda y la recompensa obtenida si se está en ese estado con esa acción:

$$Q_{k+1}(s,a) = R(s,a) + \gamma \cdot \sum_{s' \in S1} P(s' \mid s,a) \cdot V_k(s,a)$$

Es importante considerar que el cálculo del valor de la matriz Q para todas las celdas no puede realizarse de una sola vez. La única opción es recorrer la matriz entera en varias iteraciones, mejorando progresivamente su contenido hasta alcanzar la convergencia hacia una matriz óptima.

Tras rellenar todas las celdas en la matriz Q, se puede obtener el denominado vector de valor o función de valor, que representa a la función V: S -> R tal que V(s) es el máximo valor almacenado en Q (s, :), es decir, el valor máximo esperado si estamos en el estado s y escogemos la acción más favorable:

$$V_k(s) = \max_a Q_k(s,a)$$

Para considerar la convergencia del método *Value Iteration*, se puede ejecutar un número determinado de iteraciones en la matriz Q, llamados pasos, utilizando la fórmula presentada en la Figura 3.6, para medir la mejora o cambio en V a través de la distancia euclídea entre los vectores V_{k+1} y V_k. Este valor lo podemos almacenar en un vector denominado *delta (d)*. Inicialmente, se realizan un número dado de iteraciones. Una vez cargado el vector *delta*, se puede observar su convergencia para determinar si el número de pasos necesarios para alcanzar un cambio mínimo tiene que ser ajustado.

```
function distancia = distancia_euc(Vector,Vector_ant)

distancia = norm(Vector-Vector_ant);

end
```

Figura 3.6 Función del cálculo euclídeo para la convergencia del *Value Iteration*.

Otra opción sería utilizar una fórmula para calcular el valor umbral de delta (ε) en lugar de determinar el número de pasos necesarios y d es la distancia euclídea que quiere uno que haya entre dos vectores V consecutivos. Sin embargo, esta opción no es la más recomendable, ya que puede requerir un mayor número de pasos y, por lo tanto, una convergencia más lenta:

$$\varepsilon = \frac{d \cdot (1-\gamma)}{2 \cdot \gamma}$$

En cualquier caso, una vez decidida la convergencia del *Value Iteration*, hay que extraer la política óptima de la matriz Q:

$$\Pi_k(s) = arg \max_a Q_k(s,a)$$

Es decir, la política óptima es aquella que escoge como acción la que maximice el valor esperado dado que estamos en el estado s. El vector Π o política es un vector de S valores en el cada uno de los cuales guarda la acción óptima para ese estado.

Una vez generada la política se recorre todo el vector y se escribe su contenido en un archivo llamado `nueva_politica` con el formato deseado, en este caso, en un archivo.csv debido a que se puede leer esta extensión sin ningún problema en *Visual Components*.

Una vez exportada la política óptima en el archivo .csv, se carga en *Visual Components*. Para hacerlo, se accede a la pestaña de modelado (véase apartado 2.1.2) y entrando en ResourceScript dentro del controlador del robot de salida modificándolo para que, en cada estado en que se encuentre el robot, se escoja la acción adecuada. Después de importar los datos, se verifica el correcto funcionamiento del TFG.

La Figura 3.7 resume el flujo de trabajo realizado a lo largo del Trabajo Fin de Grado.

Por otra parte, se crean 2 nuevas políticas: heurística y aleatoria, con el objetivo de analizar y comparar dichas políticas (véase Figura 3.7) para verificar el buen funcionamiento de la política óptima implantada.

La política heurística se trata en el capítulo 4 y su obtención en *MATLAB*, aunque el procedimiento a nivel general como se aprecia en la Figura 3.7 es el mismo tanto para el análisis de los datos tomados a partir de la primera toma aleatoria como de la implantación nuevamente en *Visual Components* dicho anteriormente.

Para la política aleatoria, no es necesario ni analizar los datos para encontrar la política aleatoria debido a su carácter aleatorio ni implantarlo en *Visual Components* y, por tanto, los datos tomados ya se consideran resultados y se analizan junto con las demás políticas en el capítulo 4.

Es importante mencionar que cada política tiene su propio archivo de la celda de fabricación con las mismas versiones para poder distinguir con qué política se está trabajando y evitar posibles problemas al insertar la nueva política.

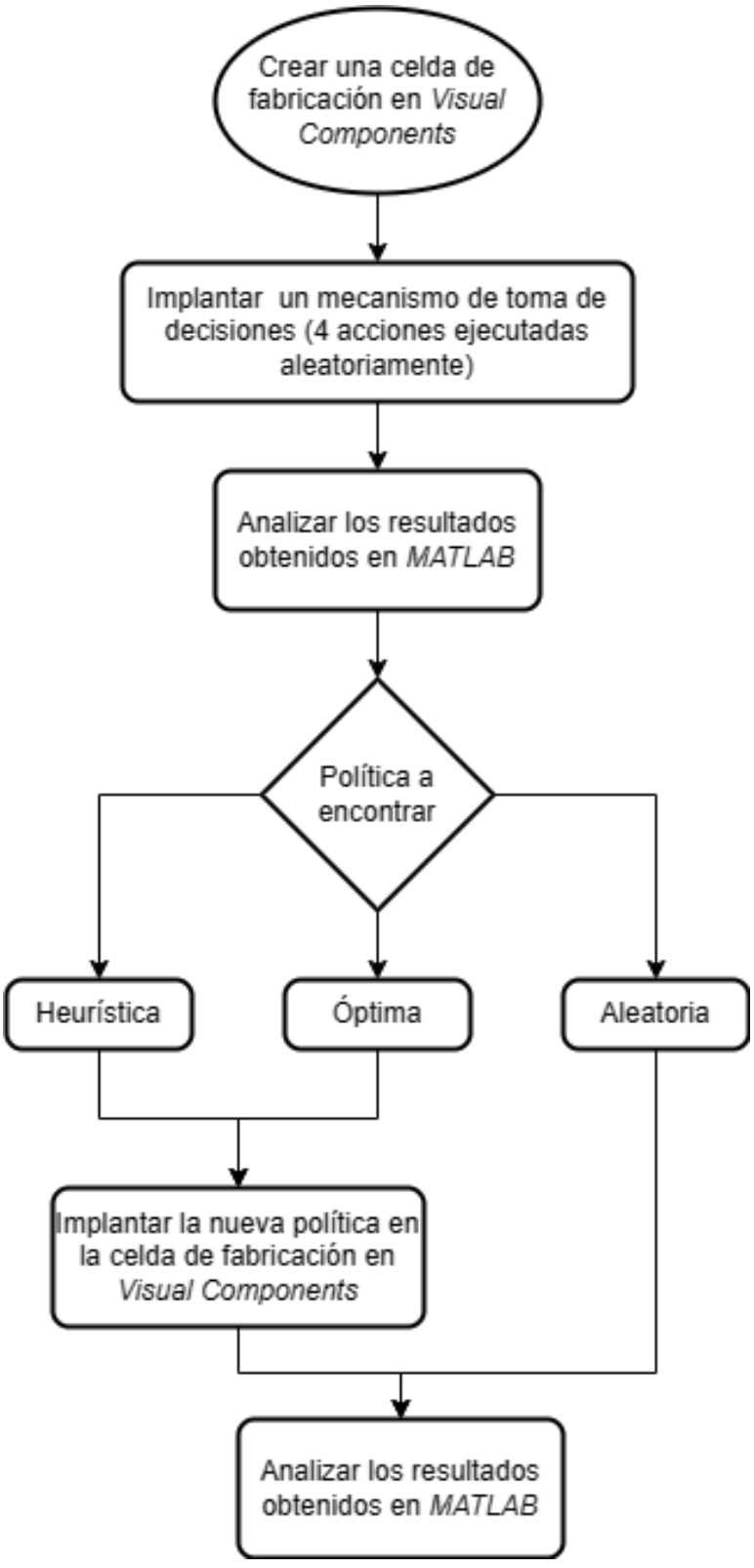


Figura 3.7: Diagrama de flujo del Trabajo Fin de Grado

Capítulo 4

Experimentación y resultados

Durante las pruebas de explotación aleatoria del entorno en *Visual Components*, se han realizado 300.000 iteraciones de simulación. Cada iteración de simulación es el número de tomas de decisiones realizadas por el robot de salida y al realizar un gran volumen de datos ha permitido rellenar adecuadamente como se refleja en la matriz proyectada de visitas. Un experimento es el conjunto de las pruebas de explotación que se ha hecho para encontrar la política óptima.
Si se desea excluir las primeras iteraciones debido a que siempre se parte de un estado inicial con cero cajas en la estantería y los robots cargados de batería, es posible hacerlo, aunque su impacto a largo plazo es mínimo. Estas pruebas han tardado aproximadamente 3 horas y se han hecho gracias a un ordenador ubicado en el laboratorio de la UMA debido al alto consumo de los recursos de la herramienta.

Una vez generada la política deseada, se implementa ésta en la celda de fabricación. A continuación, se presentan las pruebas realizadas y el análisis estadístico de algunos indicadores obtenidos a través de la modificación del código interno del programa.

4.1 Obtención de la política óptima

Tras importar en *MATLAB* los datos de exploración de las 300000 iteraciones de simulación aleatorias comentadas al inicio del capítulo 4, se genera y rellena la matriz de visitas comentada en apartado 3.2. Se muestra la matriz proyectada de visitas en la Figura 4.1.

Para ello, los datos tomados aplican la función mapear (véase Figura 3.4) para averiguar los estados posibles. Las columnas representan las acciones posibles calculadas a partir de la función de mapeo (véase Figura 3.4) y las filas los estados que pueden ser visitados.

El estado más visitado es el número 50, con un total de 15518 visitas al elegir la acción 2. Este estado corresponde a un nivel máximo de carga de la batería del robot y 9 cajas en la estantería. Esta situación puede deberse tanto a la necesidad de recargar la batería como a la mala toma de decisiones, que provoca que durante la exploración aleatoria del entorno el robot no se esté ocupando bien de sacar cajas de la estantería.

	1	2	3	4
1	34	69	58	42
2	102	248	231	101
3	28	56	56	27
4	29	67	74	20
5	27	45	40	12
6	14	40	29	18
7	15	35	25	14
8	188	392	412	179
9	1553	3079	3091	1286
10	1819	3517	3517	2170
11	122	249	255	136
12	137	282	267	136
13	110	226	220	104
14	90	188	178	88
15	56	143	113	70
16	64	125	126	79
17	44	121	135	58
18	610	1243	1324	536
19	3646	7130	7215	2912

20	6996	14260	14091	8007
21	122	237	226	130
22	138	299	302	168
23	170	316	354	175
24	155	301	297	142
25	134	264	224	125
26	100	225	246	89
27	71	146	171	84
28	653	1347	1310	579
29	3554	7213	7226	2924
30	6870	13787	13973	7626
31	33	41	53	21
32	56	111	112	56
33	116	163	182	110
34	143	255	247	118
35	184	347	343	185
36	184	339	315	167
37	199	393	412	200
38	704	1384	1483	581
39	3634	7428	7472	3148
40	6975	14042	14185	7875
41	3	6	12	6
42	9	8	15	6
43	8	22	10	6
44	4	14	18	9
45	32	59	50	31
46	50	107	124	61
47	117	245	250	129
48	531	1065	1002	478
49	2874	5787	5776	2429
50	7897	15619	15518	8035

Figura 4.1 Tabla resultante de la matriz proyectada de visitas.

Tras generar las matrices, se procede a crear el modelo de la dinámica del sistema, es decir, la matriz tridimensional de probabilidad de transición P(S1 | S0, A). El resultado de este proceso se muestra en la Figura 4.2, siendo S0 las filas y S1 las columnas, ignorando la dimensión A.

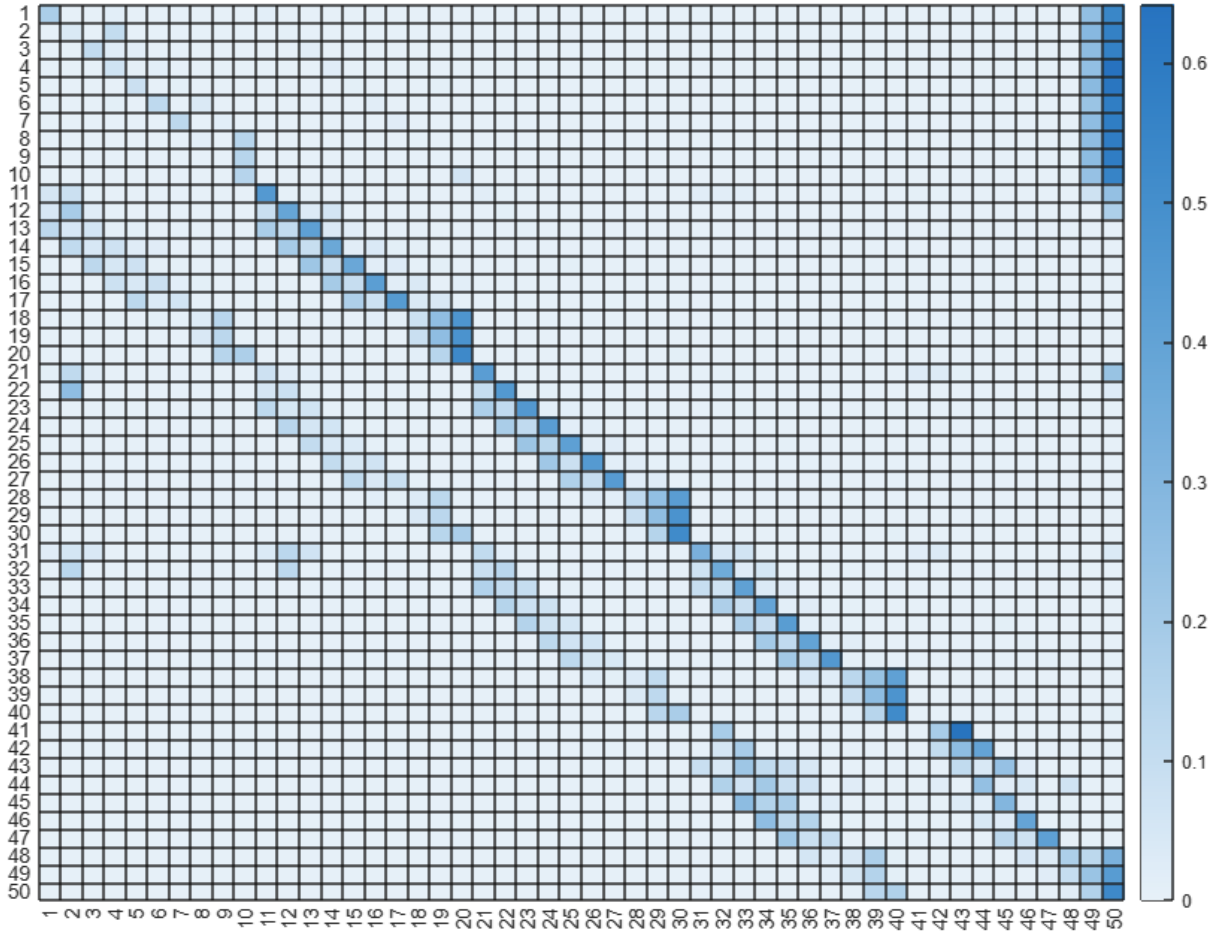


Figura 4.2 Tabla de resultados de P(S1 | S0, :).

El resultado de la probabilidad presentado en la Figura 4.2, al representar una probabilidad con un mapa de calor, el valor de cada celda se encuentra entre 0 y 1, y cuanto mayor sea, es decir, cuanto más oscuro sea el color, mayor será la probabilidad de alcanzar el estado S1 dado el S0 usando alguna acción.

Tras obtener el modelo de la dinámica del sistema mediante el proceso de convergencia, la cual se alcanza aproximadamente en unos 500 pasos de convergencia de la distancia euclídea (véase apartado 3.2), se puede visualizar en la Figura 4.3 el valor esperado acumulado con descuento obtenido para cada estado al elegir una determinada acción.

Es importante destacar que, dado que la recompensa se basa en la Tabla 3.4, cuanto mayor sea este valor, mejor será la recompensa futura. En otras palabras, el mejor escenario es estar en un estado con una acción cuyas acciones futuras tengan una recompensa de cero.

	1	2	3	4
1	-11.3834	-3.1578	-11.3834	-0.1363
2	-11.3834	-2.8521	-11.3834	-0.3553
3	-11.3834	-2.8144	-11.1816	-0.2200
4	-11.3834	-3.5702	-11.3834	-0.4265
5	-11.3834	-3.1433	-11.1439	-0.2520
6	-11.3834	-4.8375	-11.3834	-1.5612
7	-11.3834	-4.2166	-11.3834	-0.7113
8	-11.0354	-3.0085	-11.3834	-11.0270
9	-11.1868	-2.7894	-11.3834	-11.7035
10	-21.0809	-13.1174	-21.3938	-22.0716
11	-3.8962	-6.7930	-0.0592	-0.0580
12	-0.0876	-5.8764	-0.1034	-0.1083
13	-0.1571	-0.0891	-0.0937	-0.0954
14	-0.1397	-0.2023	-0.2309	-0.1915
15	-0.2602	-0.1288	-0.1814	-0.1701
16	-0.2277	-0.2807	-0.9896	-0.4621
17	-0.6733	-0.2255	-0.5903	-0.5111
18	-1.9361	-1.9873	-12.0421	-12.0694
19	-2.0017	-2.0456	-12.6671	-12.6223
20	-20.2191	-14.2042	-22.5999	-22.6206
21	-0.6881	-7.7676	-0.0293	-0.0314
22	-0.0358	-1.1372	-0.0371	-0.0383
23	-0.0475	-0.0374	-0.0404	-0.0375
24	-0.0510	-0.0546	-0.0590	-0.0544
25	-0.0748	-0.0510	-0.0612	-0.0536
26	-0.1030	-0.0866	-0.2187	-0.1671
27	-0.2023	-0.1233	-0.3967	-0.3678
28	-1.3531	-1.2914	-11.0683	-10.7904
29	-1.4126	-1.4067	-12.1384	-12.0999
30	-20.1992	-13.6077	-22.3423	-22.3237
31	-0.1243	-1.4602	-0.0199	-0.0209
32	-0.0224	-0.1632	-0.0228	-0.0222
33	-0.0270	-0.0242	-0.0279	-0.0246
34	-0.0287	-0.0264	-0.0358	-0.0272
35	-0.0418	-0.0332	-0.1051	-0.0360
36	-0.0398	-0.0452	-0.0922	-0.0576
37	-0.0644	-0.0413	-0.1514	-0.0626
38	-0.8044	-0.7779	-9.9455	-9.6457

39	-0.9403	-0.9016	-11.5196	-11.4967
40	-19.8793	-13.1688	-21.9200	-21.9045
41	-0.0191	-0.0186	-0.0189	-0.0191
42	-0.0195	-0.0204	-0.0192	-0.0194
43	-0.0226	-0.0218	-0.0212	-0.0255
44	-0.0216	-0.0695	-0.0873	-0.0263
45	-0.0234	-0.0294	-0.0245	-0.0245
46	-0.0285	-0.0232	-0.1196	-0.0423
47	-0.1259	-0.0265	-0.0282	-0.0300
48	-0.4548	-0.4702	-7.9804	-7.9073
49	-0.5931	-0.5305	-10.4476	-10.1732
50	-18.8519	-12.6482	-21.4616	-21.4531

Figura 4.3 Tabla Q resultante del aprendizaje.

Finalmente, con toda la información generada anteriormente, es posible crear la política deseada mediante un vector que almacena, para cada estado correspondiente a una fila, la acción óptima a tomar (véase Figura 4.4).

1	4
2	4
3	4
4	4
5	4
6	4
7	4
8	2
9	2
10	2
11	4
12	1
13	2
14	1
15	2
16	1
17	2
18	1
19	1
20	2
21	3
22	1
23	2
24	1
25	2
26	2
27	2
28	2
29	2
30	2
31	3
32	4
33	2
34	2
35	2
36	1
37	2
38	2
39	2
40	2
41	2
42	3
43	3
44	1
45	1
46	2
47	2
48	1
49	2
50	2

Figura 4.4 Tabla resultante del vector de política óptima.

En el apartado 4.4 se lleva a cabo un análisis más detallado de los resultados obtenidos con esta política.

4.2 Obtención de la política aleatoria

Usar una política aleatoria de acción (es decir, la misma que se utiliza para explorar el entorno y construir el modelo de su dinámica) es útil como línea base de comparación del rendimiento de otras políticas.

Para garantizar una adecuada implementación de esta política aleatoria es posible obtener los datos directamente desde *Visual Components* tal y como se muestra en la Figura 3.3. Con este fin, se programa en Python la variable `Poli_num` de manera que, en cada iteración, pueda asignársele un valor aleatorio dentro del rango de acciones permitidas. Es posible obtener y examinar la política a través de las muestras presentadas en la sección 4.4, donde se expondrán los estadísticos correspondientes.

4.3 Construcción de la política heurística

En este trabajo se ha estudiado la ventaja de integrar inteligencia artificial en un entorno de simulación industrial. Para ello se compara la política óptima obtenida por la primera con una implementada directamente por una persona, que denominamos política heurística.

Esta política implica evaluar el estado actual del robot y, en función de ello, tomar una acción determinada de manera intuitiva. Este proceso es manual y no requiere el uso de fórmulas matemáticas ni la consideración de estados futuros, sino que se basa en el criterio propio de la autora de este trabajo.

En la Figura 4.5, la primera columna muestra los estados posibles, la segunda el nivel de batería asociado a cada estado, y la tercera el número de cajas presentes en cada estado. Esto se consigue gracias a la extracción al principio de 300000 muestras aleatorias comentadas al inicio del capítulo 4, se procesan gracias a la función mapear (véase Figura 3.4) los estados y a partir de este, desmapear (véase Figura 3.8) para averiguar tanto el nivel de batería como el recuento de cajas en la estantería.

	1	2
1	1	0
2	1	1
3	1	2
4	1	3
5	1	4
6	1	5
7	1	6
8	1	7
9	1	8
10	1	9
11	2	0
12	2	1
13	2	2
14	2	3
15	2	4
16	2	5
17	2	6
18	2	7
19	2	8
20	2	9
21	3	0
22	3	1
23	3	2
24	3	3
25	3	4
26	3	5
27	3	6
28	3	7
29	3	8
30	3	9

31	4	0
32	4	1
33	4	2
34	4	3
35	4	4
36	4	5
37	4	6
38	4	7
39	4	8
40	4	9
41	5	0
42	5	1
43	5	2
44	5	3
45	5	4
46	5	5
47	5	6
48	5	7
49	5	8
50	5	9

Figura 4.5 Tabla de estados y sus descripciones discretizadas.

Se puede observar que cada 10 estados se repiten la secuencia del número de cajas. Teniendo en cuenta tanto eso y el nivel de batería como la capacidad de la estantería, se pueden presentar varios casos y, en función de ellos, tomar la acción correspondiente:

- Si el estado tiene más de una caja en la estantería, la decisión más lógica sería llevarse ambas cajas, teniendo en cuenta que el nivel de batería puede ser muy bajo, como se mencionó en el punto 3.1.1.
- Si solo hay una caja, el robot solo recogerá una caja.
- Si no hay cajas en la estantería, el robot se tomará un descanso de 50 segundos.
- Si no se cumple ninguna de las condiciones anteriores, el robot se recargará.

Para generar el vector de política en *MATLAB*, se ha implementado esta lógica (véase Figura 4.6) en una función que la usa para rastrear todos los estados y almacenar la acción heurística correspondiente.

```
for f = 1:Nest
    Tabla_aux(f,1) = f;
    if (Tabla_aux(f,1) > 42 || (Tabla_aux(f,1) > 32 && Tabla_aux(f,1) < 41) || (Tabla_aux(f,1) > 22 && Tabla_aux(f,1) < 31) || (Tabla_aux(f,1) > 12 && Tabla_aux(f,1) < 21)) % 2. Coger dos cajas
            Politica(f) = 2;
            Tabla_aux(f,2)=2;
    elseif (Tabla_aux(f,1) == 42 || Tabla_aux(f,1) == 32 || Tabla_aux(f,1) == 22 || Tabla_aux(f,1) == 12)% 1. Coger una caja
            Politica(f) = 1;
            Tabla_aux(f,2)=1;
    elseif (Tabla_aux(f,1) == 41 || Tabla_aux(f,1) == 31 || Tabla_aux(f,1) == 21 || Tabla_aux(f,1) == 11) % 3. Reposar
            Politica(f) = 3;
            Tabla_aux(f,2)=3;
    else      % 4.Recargar
        Politica(f) = 4;
        Tabla_aux(f,2)=4;
    end
end
```

Figura 4.6 Código generado para la toma de decisiones en la política heurística.

En la Figura 4.7 se muestra una tabla en la que la primera columna representa el estado y la segunda columna indica la acción seleccionada según el criterio previamente mencionado (véase Figura 4.6).

1	4
2	4
3	4
4	4
5	4
6	4
7	4
8	4
9	4
10	4
11	3
12	1
13	2
14	2
15	2
16	2
17	2
18	2
19	2
20	2
21	3
22	1
23	2
24	2
25	2
26	2
27	2
28	2
29	2
30	2
31	3

32	1
33	2
34	2
35	2
36	2
37	2
38	2
39	2
40	2
41	3
42	1
43	2
44	2
45	2
46	2
47	2
48	2
49	2
50	2

Figura 4.7 Política heurística.

Esta nueva política se ha incorporado al Trabajo Fin de Grado y se ha realizado un análisis detallado de los resultados obtenidos, como se verá en el siguiente apartado.

4.4 Resultados estadísticos

Los indicadores de eficiencia que se evaluarán en este estudio se conocen como *KPI* (Indicadores Clave de Rendimiento)[18]. Estos indicadores se basan en un modelo que divide las diferentes etapas que se llevan a cabo en un almacén. Dado que no existe un consenso establecido sobre qué *KPI* utilizar, tenemos cierta libertad para elegir los más adecuados para nuestro caso. Los primeros indicadores se enfocan en el rendimiento del robot, mientras que los demás se centran en la eficiencia de la gestión del almacén.

Para llevar a cabo este estudio, se han realizado 100 mediciones de *KPI* a lo largo de 24 horas de simulación, con un intervalo de 60 segundos de simulación entre cada medición, siguiendo el horario simulado del programa. Los datos recopilados se han

procesado utilizando *MATLAB*, automatizando la apertura y el almacenamiento de los archivos en una matriz. Esta matriz se caracteriza por tener el número de muestras tomadas como columnas y el intervalo de tiempo entre cada medición como filas. En otras palabras, para cada fila que representa un intervalo de 60 segundos, se han recogido 100 mediciones.

Se ha decidido utilizar la mediana como medida de tendencia central de los indicadores, ya que representa el valor central en un conjunto de datos ordenados de menor a mayor. Esta medida solo se puede calcular para variables cuantitativas. En situaciones donde el tamaño de la muestra es pequeño o donde hay valores atípicos, la mediana es preferible a la media como indicador de la tendencia central, ya que es menos sensible a valores extremos y proporciona una representación más precisa de la tendencia central de los datos en estos casos [19].

En los siguientes subapartados se presentan los resultados estadísticos obtenidos para cada indicador según las distintas políticas de funcionamiento del almacén aplicadas.

4.4.1 Distancia recorrida por el robot

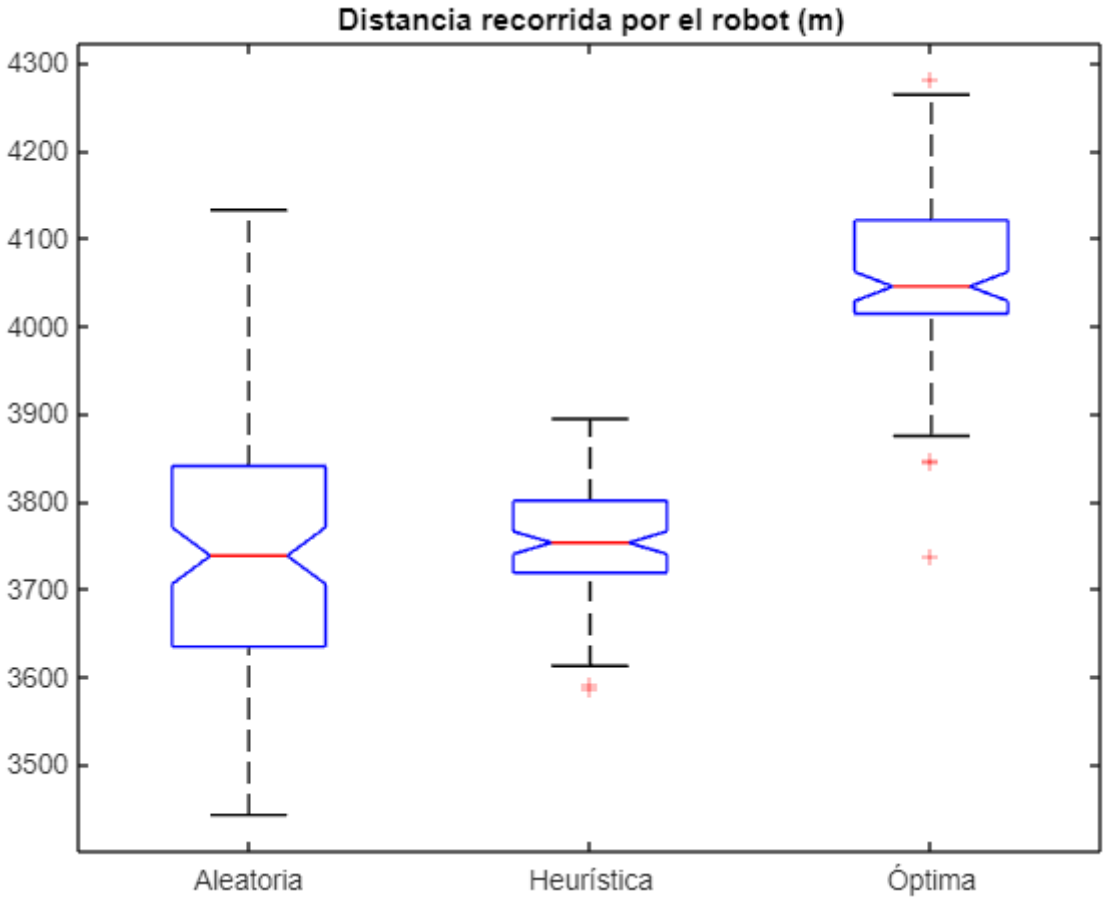


Figura 4.8 Distancia recorrida del robot en metros.

Se ha analizado la distancia recorrida por el robot en metros, agrupando los datos de cada muestra de cada política para obtener la distancia final recorrida. Como se puede apreciar, la política aleatoria varía en un intervalo entre 3650 m y 3850 m debido a su naturaleza aleatoria. Al observar la mediana de las políticas heurística y óptima, se puede ver que la distancia recorrida en la política óptima es considerablemente mayor que en la heurística.

Por lo tanto, se deduce que la política óptima es la que ha hecho que el robot recorra más distancia. Recuérdese que la política óptima persigue minimizar la probabilidad de que la estantería se llene por completo (lo cual es penalizado). Por lo tanto, que eso lleve a recorrer más distancia es lógico si tenemos en cuenta que el robot debe

moverse más veces para mantener la estantería por debajo de su ocupación máxima ya sea porque en el estado que este coge 1 caja en vez de 2 o ya sea porque estaba descansando o recargando y se le han acumulado las cajas.

4.4.2 Tiempo de recarga del robot

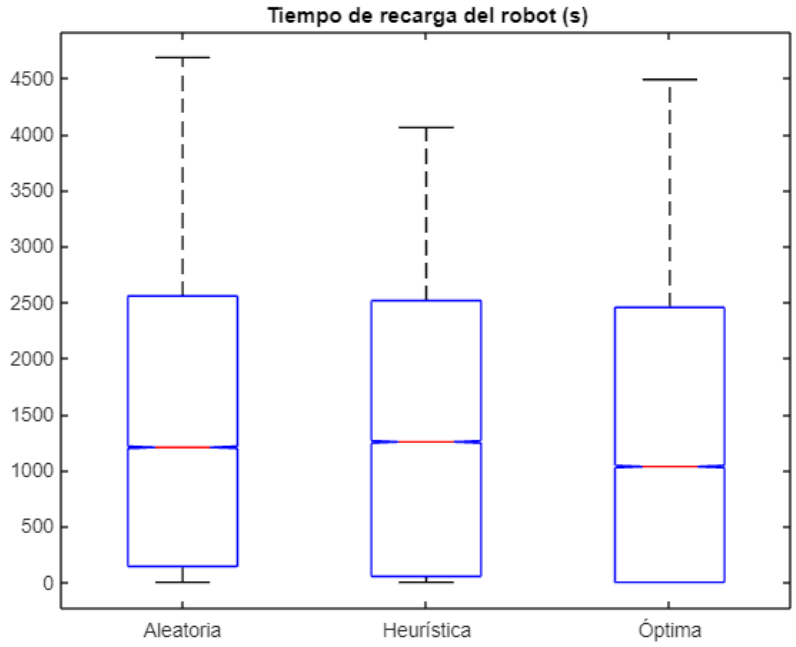


Figura 4.9 Tiempo de recarga del robot en segundos.

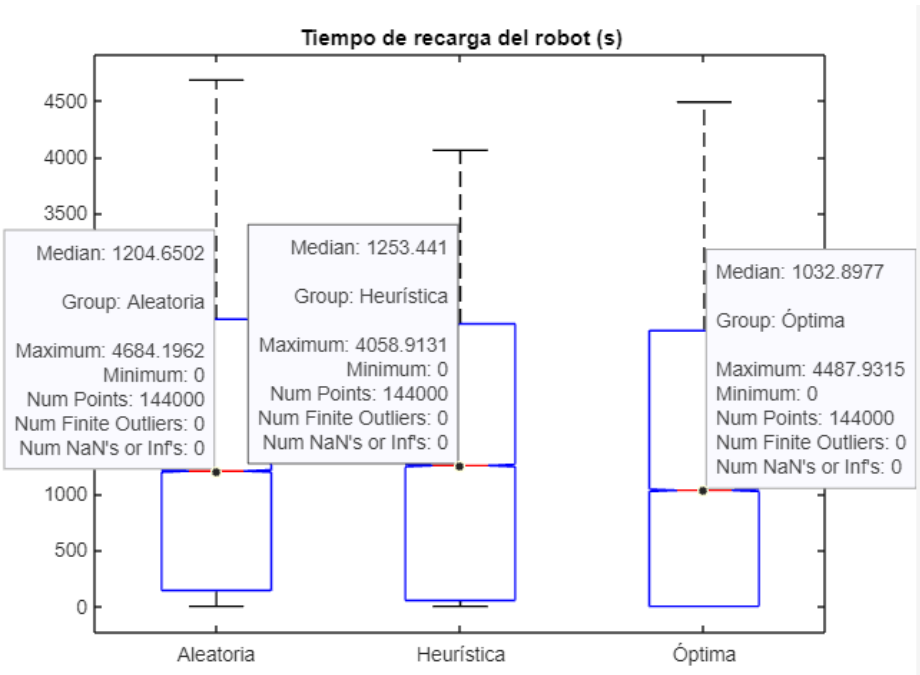


Figura 4.10 Mediana del tiempo de recarga del robot en segundos.

En la Figura 4.9 se muestra el tiempo de recarga del robot para las 3 políticas. Al analizar más detalladamente la Figura 4.10, se puede observar que las tres políticas son muy similares en cuanto al tiempo de recarga del robot. Al observar la mediana, parece que la política óptima es la mejor, seguida de la política aleatoria, pero las diferencias estadísticas parecen irrelevantes. Esto es lógico, ya que el tiempo que tarda en recargarse el robot es habitualmente el mismo.

Es importante recordar, como se menciona en el apartado 3.1, que en todas las políticas se tiene en cuenta el hecho de que, si el nivel de batería del robot es inferior al 20%, éste debe ir a recargarse para evitar quedarse sin energía. Esto afecta a la recarga de la batería, ya que, en un espacio con un dimensionado de la celda más grande, el robot tendría que recorrer más distancia y, por tanto, recargar más veces. Esto podría dar lugar a situaciones en las que, con un 20% de batería, el robot se quede corto y se produzca un cambio significativo en los tiempos de recarga y en el número de políticas.

4.4.3 Número de veces que el robot recarga

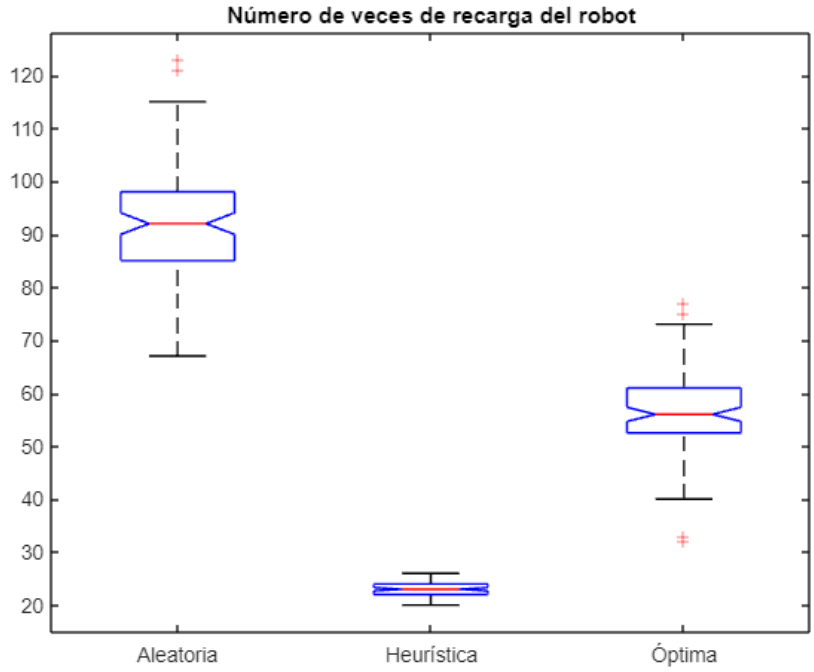


Figura 4.11 Números de veces de recarga del robot.

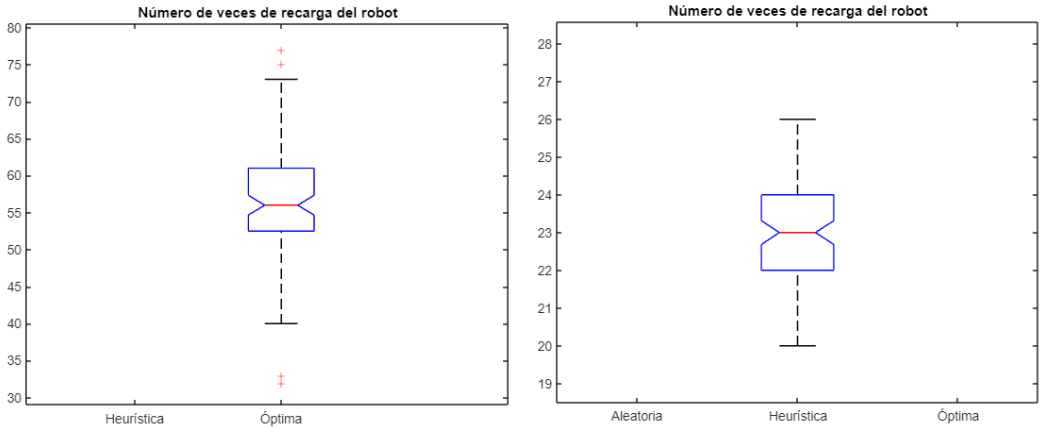


Figura 4.12 Números de veces de recarga del robot, ampliada.

Se ha analizado el número de veces que el robot se recarga, agrupando los datos de cada muestra de cada política para obtener el número final de veces que se recarga(véase Figura 4.11). Al observar la gráfica, se puede ver que la política heurística es la mejor en este caso, ya que el robot se recarga muy pocas veces, mientras que en la política aleatoria es la que más tarda debido a que, como se ha visto en el apartado 4.3, en la política heurística solo se recarga cuando el nivel de batería es 1(véase Figura 3.2).

Esta es la medición menos dudosa ya que la política heurística esta pensada muy bien que el robot recargue poco (no es el objetivo de la celda de fabricación)

4.4.4 Colisiones con el humano

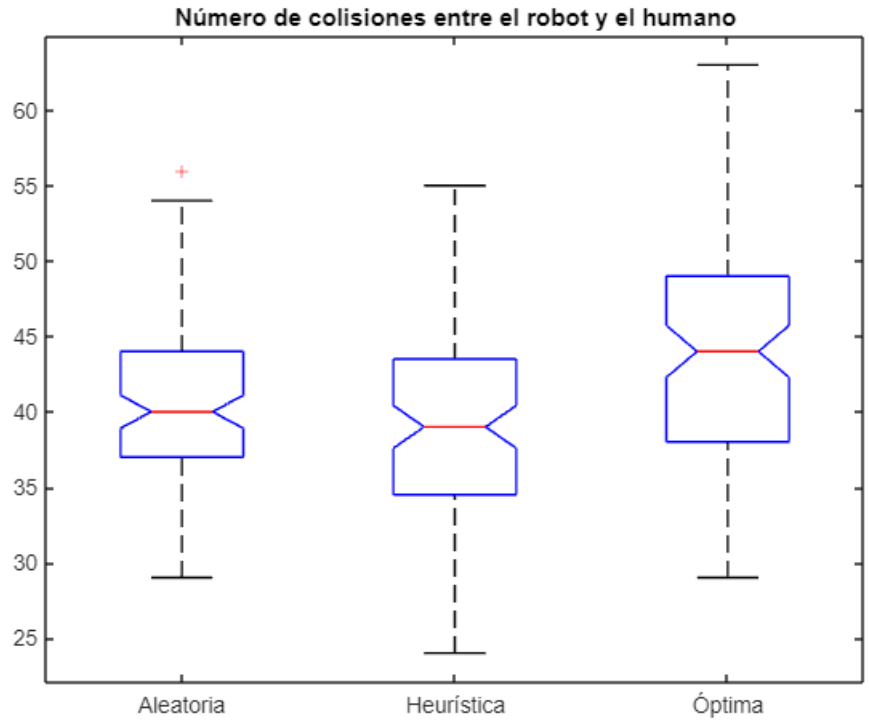


Figura 4.13 Numero de colisiones entre el robot y el humano.

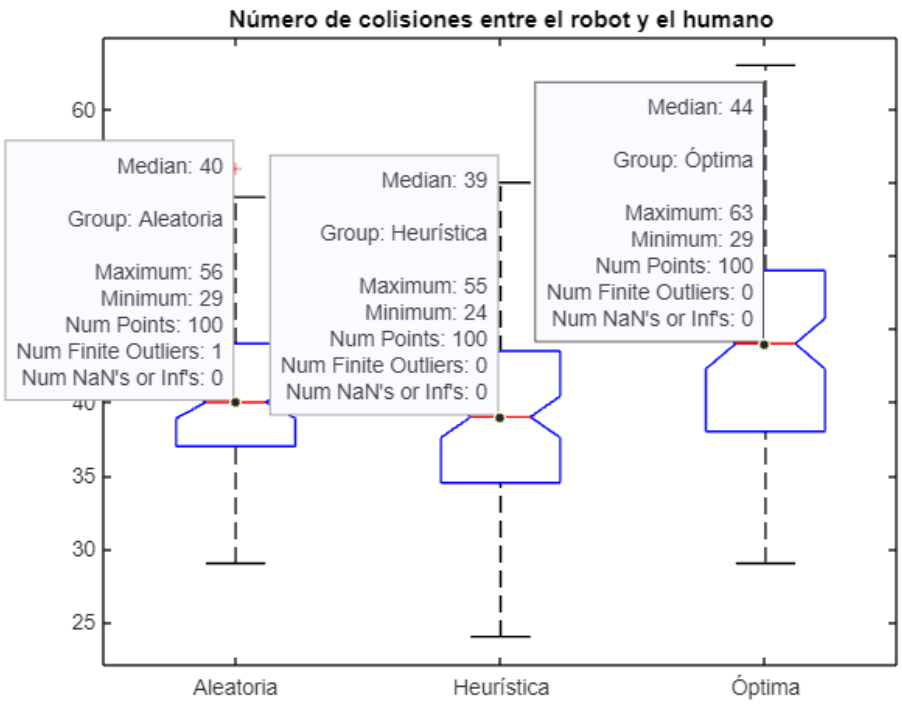


Figura 4.14 Mediana del número de colisiones entre el robot y el humano.

Se ha llevado a cabo un análisis del número de colisiones entre humanos y robots, así como del tiempo transcurrido entre ellas en segundos. Los datos se han organizado por muestra y política para obtener el recuento final de colisiones y el tiempo entre ellas. Al examinar la Figura 4.14, se puede observar que la política óptima presenta el peor resultado en términos de número de colisiones, ya que tiene la mayor cantidad, mientras que la política heurística tiene la menor cantidad.

En cuanto al tiempo entre colisiones en la Figura 4.16 , la política heurística es la peor, ya que hay menos tiempo entre una colisión y la siguiente, mientras que la política óptima es mejor en este aspecto, ya que permite un mayor margen de tiempo entre colisiones al tener una dispersión ligeramente mejor, aunque es comparable.

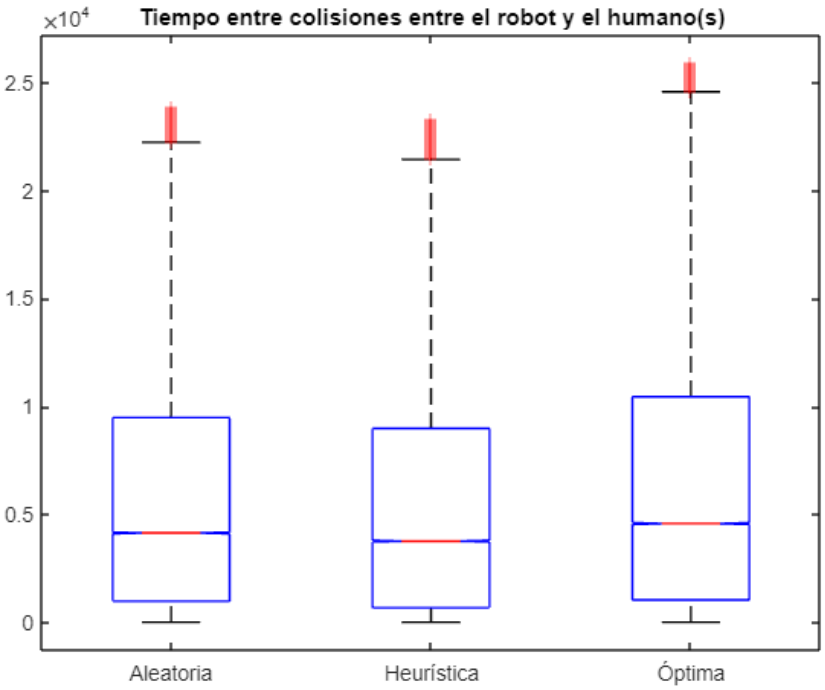


Figura 4.15 Tiempo entre colisiones entre el robot y el humano en segundos.

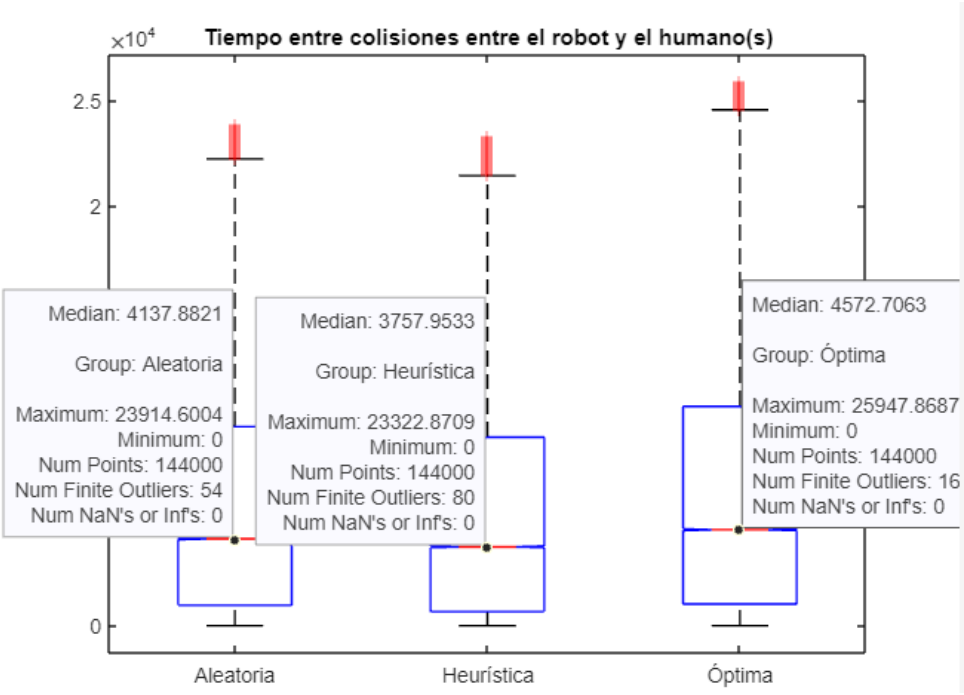


Figura 4.16 Mediana de los tiempos entre colisiones entre el robot y el humano en segundos.

4.4.5 Número total de cajas servidas

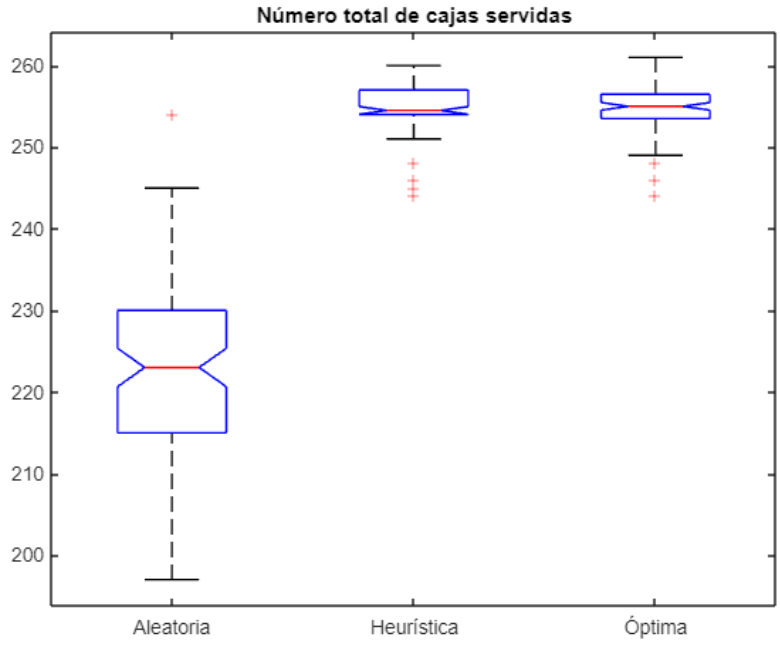


Figura 4.17 Número total de cajas servidas.

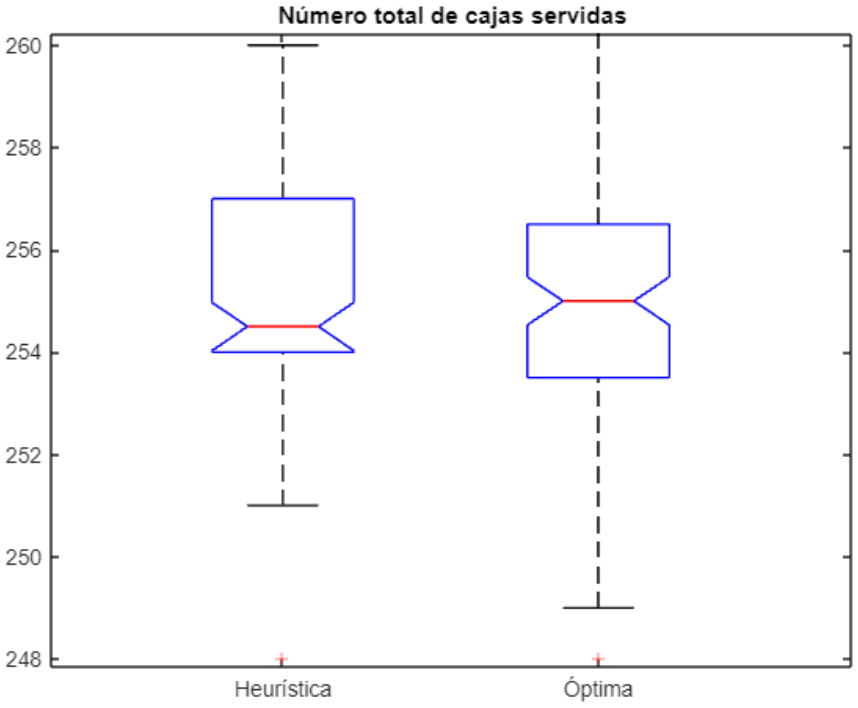


Figura 4.18 Número total de cajas servidas, ampliada.

Se ha llevado a cabo un análisis del número total de cajas servidas, organizando los datos de cada muestra y política para determinar el número total de cajas que han salido de la celda de fabricación(véase Figura 4.17 y Figura 4.18).

A partir de este análisis, se puede concluir que la política óptima es ligeramente mejor que la política heurística propuesta ya que es capaz de dar salida a un mayor número de cajas y por tanto es lógico ya que se está cumpliendo el objetivo de la política óptima minimizar la probabilidad de que la estantería se llene por completo haciendo que salgan un mayor número de cajas del almacén. Por otra parte, también es normal acerca de la poca diferencia entre la política óptima y la heurística ya que está viendo la política que ha generado (véase Figura 4.7) tiene como objetivo recoger 2 cajas mayoritariamente.

4.4.6 Tiempo de espera de una caja en la estantería

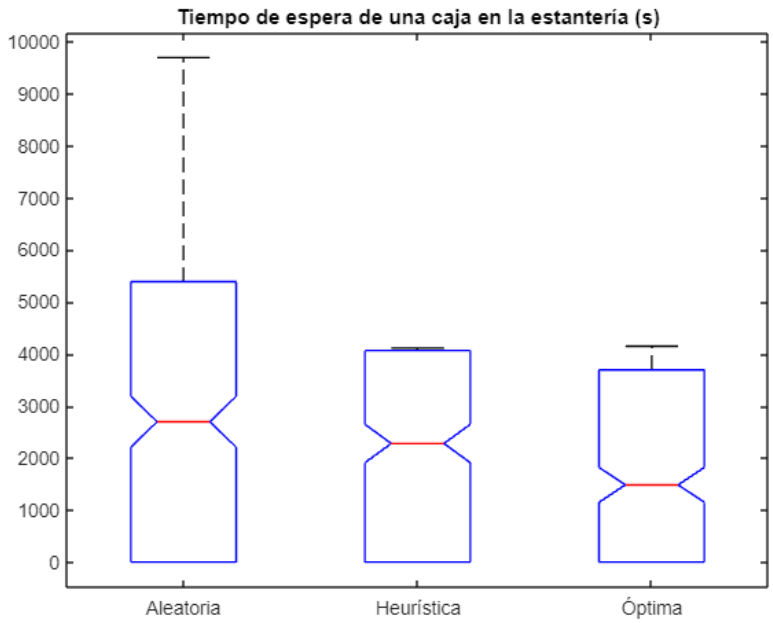


Figura 4.19 Tiempo de espera de una caja en la estantería en segundos.

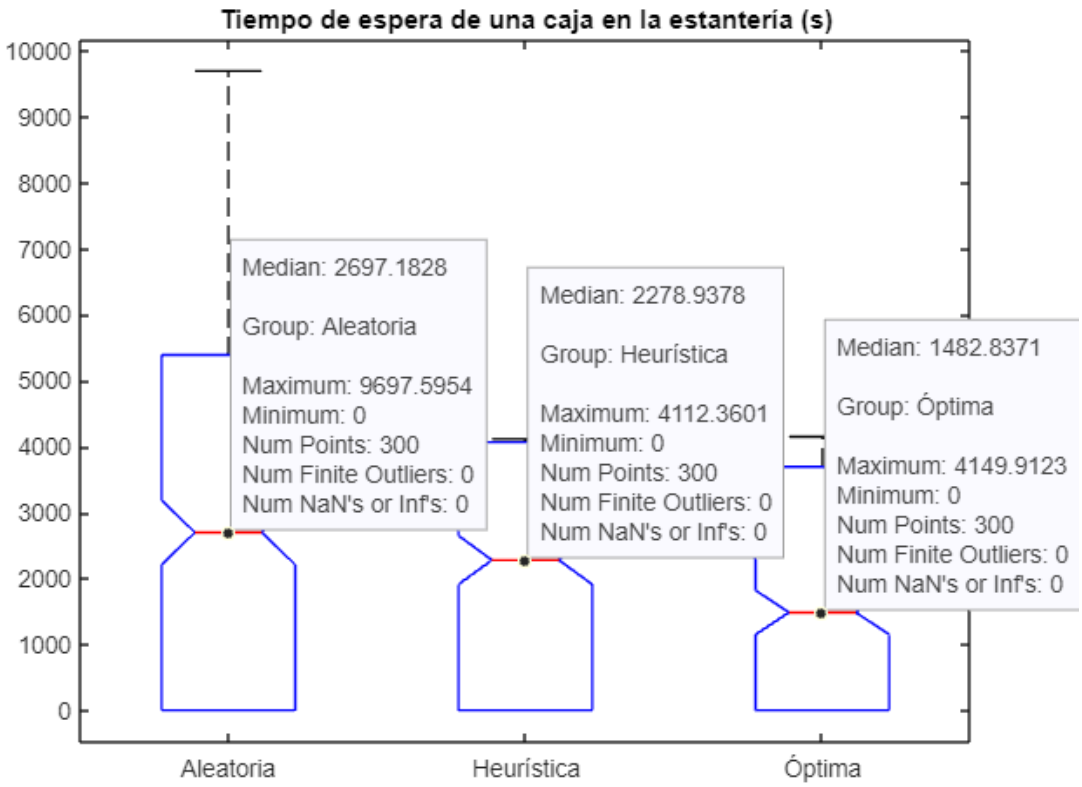


Figura 4.20 Mediana del tiempo de espera de una caja en la estantería en segundos.

Al examinar la muestra, se observa que el tiempo promedio de espera en segundos (véase Figura 4.20) de una caja es muy similar entre las políticas heurística y aleatoria. Dado que no se desea llenar completamente la estantería, la política óptima es aquella que asegura un tiempo de espera reducido para cada caja y fijándose en la Figura 4.19 se observa que, aunque en la varianza y en rango es más o menos parecida, pero con la mediana se observa que se comporta mejor.

4.4.7 Tiempo de espera de una caja desde que se introduce en la estantería hasta que sale del almacén

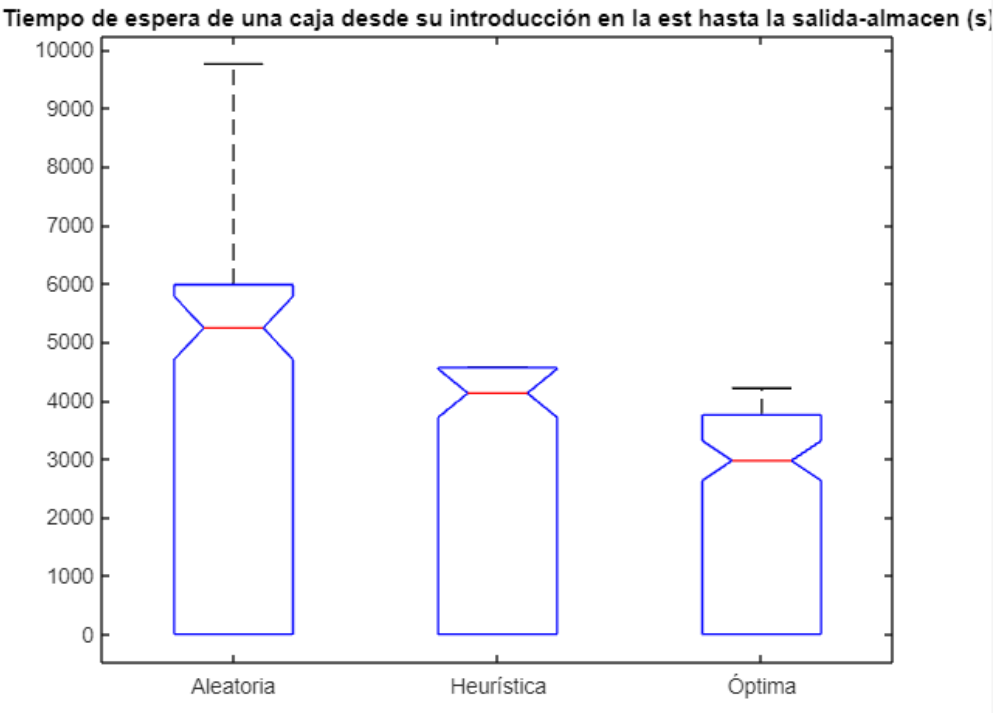


Figura 4.21 Tiempo de espera de la caja desde que se introduce en la estantería hasta su salida en segundos.

En esta muestra se ha analizado el tiempo de espera de una caja desde su introducción en la estantería hasta su salida de la celda de fabricación. La política aleatoria resulta ser la peor opción, ya que presenta el valor más alto en la mediana del tiempo de espera. Por otro lado, la política óptima es mejor que la heurística, ya que las cajas pasan algo menos de tiempo en el almacén e incluso en el peor de los casos de esta, está por debajo de la mediana de la heurística y la dispersión es ligeramente menor.

4.4.8 Número de veces que se llena la estantería completamente

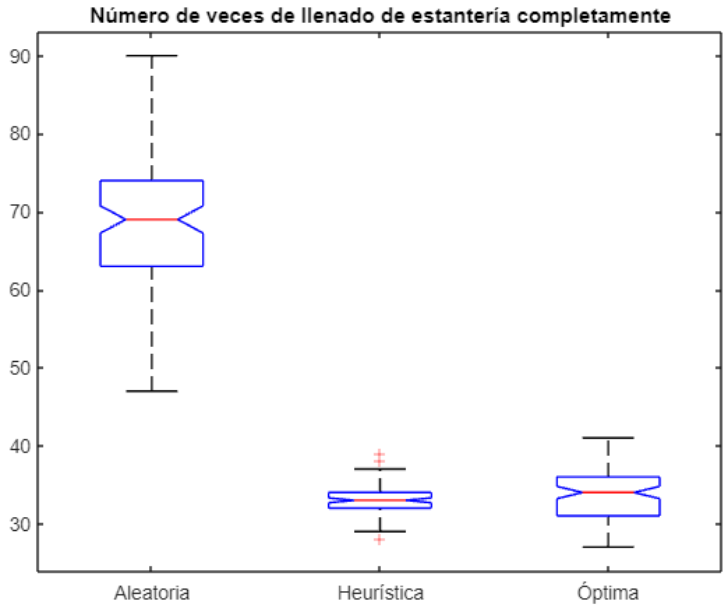


Figura 4.22 Número de veces de llenado de la estantería.

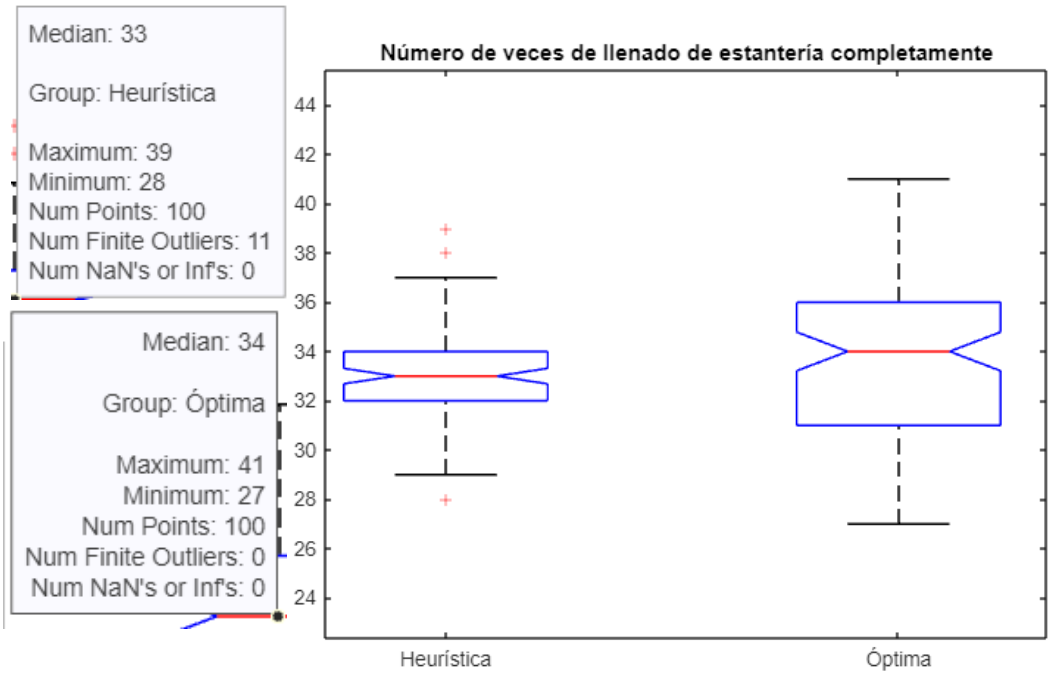


Figura 4.23 Número de veces de llenado de la estantería, ampliada.

Al analizar la muestra, se puede observar que la política aleatoria es la que más llena la estantería por completo, mientras que tanto la política heurística como la óptima son aproximadamente comparables porque la mediana es muy parecida, pero tiene menos dispersión(la varianza es más pequeña en la heurística). Es comparable, significa, como la autora, siendo persona humana, al añadir más información se ha conseguido reducir la varianza de la política óptima de la mayoría de las muestras, pero realmente el comportamiento de la heurística es muy parecido a la óptima. La política heurística es tan buena porque tiene más información y se está usando eso haciendo que no solo se analiza el llenado de la estantería, sino que está mirando varios *KPI* más.

La política heurística al crearla de una manera tal que aparte de querer dar salida a las cajas, se recarga solo cuando se está con un nivel de batería muy bajo hace que la heurística mejore como se ha visto en el aspecto de la recarga de batería junto con el llenado de estantería y salida de las cajas en el almacén.

La solución que encuentra la política óptima al tener el único objetivo de no llenar la estantería completamente (véase Tabla 3.4) se centra en este único *KPI,* pero al hacerlo consigue bajar otros porque son dependientes y ,por tanto, consigue mejorar varios de los *KPI* vistos anteriormente. Además, hay que recordar que lo ha encontrado *Value Iteration*.

No se puede mejorar mucho o nada la política óptima salvo la varianza que tenga (*Value Iteration* nunca busca que la varianza sea mejor pero sí que busca la media) y por tanto, el heurístico con toda esa información extra de conocimiento humano consigue ser óptimo y no está mal ni pone en mal lugar a *Value Iteration*.

En resumen, lo que se ha conseguido con la política heurística basándose en conocimiento humano que el *Value Iteration* no tiene, se ha conseguido la misma optimalidad que consigue el *Value Iteration*.

4.4.9 Análisis de los resultados estadísticos

En la Tabla 4.1 se puede observar una recopilación para cada *KPI* su mejor política en negrita, ya sea tomando el valor de la mediana o la media, ya que se ha creado esa tabla así debido a que la varianza es muy parecida en la mayoría de los casos. De acuerdo con los análisis de estadísticos presentados, se pueden extraer varias conclusiones.

La política heurística, se ha diseñado con el objetivo de cargar menos y tener menos cajas en el almacén y se ha visto reflejado en el número de veces de recarga, y el número de llenado de la estantería, aunque tanto el número total de cajas servidas como los tiempos de espera del producto en la celda son menos eficientes.

Como se ha comentado anteriormente, el único objetivo de la política óptima es en no llenar completamente la estantería (véase Tabla 3.4). También al tener otras medidas *KPI* que no son los que la óptima buscaba, sin embargo, en la mayoría de ellas, por número, la política óptima se ha visto ligeramente mejor que la heurística, pero no es el objetivo de la óptima. Sin embargo, la heurística en los otros sí que buscaba recargar poco y el número total de cajas servidas porque se ha definido así y a pesar de buscarlo, la óptima ha conseguido un comportamiento bueno.

Finalmente, en la política aleatoria, al ser aleatoria, hay muestras en las que resulta eficiente, aunque no la mejor, pero en la mayoría resulta es la peor. En general, produce mucha más varianza (incertidumbre) en todos los indicadores.

MEJOR POLÍTICA PARA CADA KPI	DISTANCIA RECORRIDA POR EL ROBOT (M) MEDIANA	TIEMPO DE RECARGA DEL ROBOT (S) MEDIANA	NÚMERO DE VECES DE RECARGA DEL ROBOT MEDIANA	TIEMPO ENTRE COLISIONES ENTRE EL ROBOT Y EL HUMANO (S) MEDIANA	NÚMERO DE COLISIONES ENTRE EL ROBOT Y EL HUMANO MEDIANA	NÚMERO TOTAL DE CAJAS SERVIDAS MEDIANA	TIEMPO DE ESPERA DE UNA CAJA EN LA ESTANTERÍA (S) MEDIANA	TIEMPO DE ESPERA DE UNA CAJA DESDE QUE SE INTRODUCE EN LA ESTANTERÍA HASTA QUE SALE DEL ALMACÉN (S) MEDIANA	NÚMERO DE VECES DE LLENADO DE LA ESTANTERÍA MEDIA
Óptima	**4045**	**1032.897**	56	**4572.706**	44	**255**	**1482.837**	**29687**	33.462
Heurística	3753	1253.441	**23**	3757.953	**39**	254	2278.938	41237	**33.410**
Aleatoria	3738	1204.650	92	4137.882	40	223	2697.183	52391	68.051

Tabla 4.1 Resumen de las mejores políticas para cada *KPI*.

Capítulo 5

Conclusiones y futuros trabajos

5.1 Conclusiones

En resumen de todo lo que se ha logrado, se ha implementado una celda de fabricación simulada que reproduce una línea de producción automática con múltiples procesos de trabajo coordinados y conectados. Se ha integrado la herramienta *Visual Components* y con inteligencia artificial para la toma de decisiones óptimas. Se ha establecido un mecanismo para recopilar datos y exportarlos para su análisis y optimización. Los resultados obtenidos se han estudiado de cara a mejorar la toma de decisiones en la celda y optimizar los procesos de producción y el uso eficiente de los recursos.

En relación con los objetivos establecidos tanto en el anteproyecto como en el capítulo 1, se puede afirmar que se han logrado todos ellos. Se ha realizado un análisis exhaustivo de la viabilidad de *Visual Components* y se ha implementado con éxito un algoritmo de inteligencia artificial para optimizar el proceso. En el marco del TFG, se ha desarrollado un mecanismo de toma de decisiones para el robot de salida, como se detalla en la Tabla 3.1. Los datos obtenidos de la simulación se han procesado en *MATLAB* para estimar un modelo dinámico con incertidumbre y determinar la política óptima que maximiza el uso de los recursos disponibles (véase

apartado 4.1). Además, se han analizado los datos con dos políticas adicionales: una determinada por una persona y otra completamente aleatoria.

Dificultades técnicas

Como se menciona en el apartado 2.1, aunque hay documentación y ayuda disponible para entender el funcionamiento de *Visual Components*, la implementación de lo aprendido ha sido un desafío considerable. Esto se debe a que esta herramienta no fue desarrollada inicialmente para fines de aprendizaje por refuerzo, sino que está más orientada a la simulación. Por lo tanto, la información necesaria para la creación de la celda de fabricación, como documentación y tutoriales en video, no está estructurada para este propósito. Esto hace que sea necesario recopilar información de diferentes fuentes y agruparla sin romper el TFG. Por ejemplo, es necesario integrar *Python*, combinar estadísticas y extraerlas, trabajar con *MATLAB* para encontrar las políticas y volver a pasarlas a *Visual Components*.

Un aspecto menos importante pero que también merece ser mencionado es que se han encontrado dificultades al actualizar *Visual Components* de la versión 4.3 a la 4.7. Los componentes de la versión anterior tenían métodos de creación y manipulación diferentes, lo que ha hecho que el trabajo realizado en la versión anterior no sea compatible con la nueva. Además, para trabajar con *Visual Components*, es necesario el uso de una VPN (consulte el Apéndice A, *Manual de utilización de la celda de fabricación y el entorno de análisis*). Sin embargo, hay que tener en cuenta que la VPN se desconecta cada 2 horas, lo que requiere una rápida reconexión, especialmente durante pruebas que duran más de dos horas.

5.2 Trabajos futuros

Como se ha mencionado anteriormente, es posible ampliar el alcance de la aplicación a escenarios más complejos, utilizando herramientas de toma de decisiones más avanzadas. Estas herramientas podrían incluir recompensas positivas y otras más sofisticadas, e incluso podrían estar enfocadas a diferentes tipos de condiciones. Al utilizar estas herramientas más avanzadas, se podrían abordar problemas más desafiantes y obtener soluciones más precisas y eficientes en la toma de decisiones.

También se podría trabajar con una discretización de estados más fina (más estados) para ver si influye mucho o poco en la calidad de la política óptima obtenida. Otros cambios pueden aplicarse al estado inicial, modificando el nivel de carga de la batería y el número de cajas presentes en la estantería al inicio del experimento, aunque su implementación puede resultar compleja. También si se cambia la recompensa dada, incluyendo por ejemplo la recarga del nivel de batería, ya que implicaría hacer otro estudio y otra comparativa.

Apéndice A

Manual de utilización de la celda de fabricación y el entorno de análisis

En este apartado se presenta una guía detallada de los elementos usados de cada herramienta para la utilización de la celda de fabricación y de la obtención de la política óptima. Esta guía tiene como objetivo proporcionar información suficiente para que el lector pueda recrear, ejecutar o modificar este Trabajo Fin de Grado.

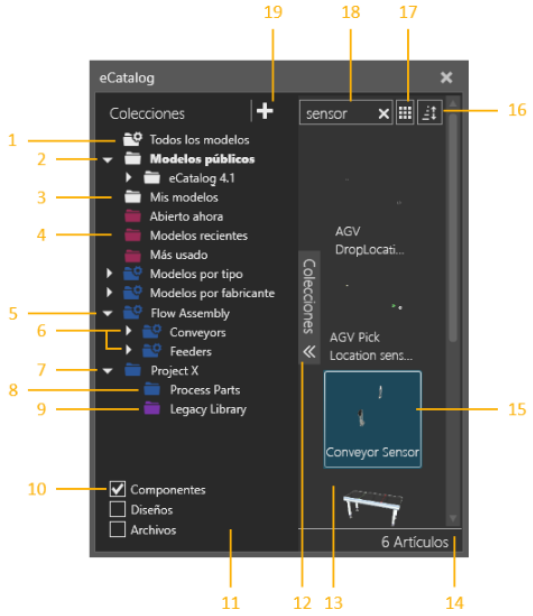
Requerimientos

Para instalar correctamente *Visual Components* como miembro de la Universidad de Málaga, se debe visitar la sección de descarga del software del SCI de la UMA [20] y seguir las instrucciones de instalación. Hay que tener en cuenta que, al proporcionar una licencia en red, el equipo utilizado debe estar conectado a la red de la Universidad de Málaga. Para lograr esto, hay que utilizar el servicio de túneles SSL [21] proporcionado por dicho servicio; una vez que se ha verificado la conectividad con la red, se procede como si el equipo estuviera conectado directamente a ella.

Para el uso correcto de *MATLAB* como miembro de la Universidad de Málaga, igualmente se debe visitar el enlace del SCI de la UMA [22] y seguir las instrucciones en la sección de instalación.

Visual Components

- *Componentes*. Para la creación de la celda de fabricación, se han extraído varios componentes del catálogo de componentes *eCatalogue*, incluyendo estantería y robots. Estos se pueden encontrar filtrando los modelos por fabricante y seleccionando *Visual Components*, donde se encuentran todos los componentes utilizados en este trabajo (véase Figura A.1).



1. Colección del sistema de todos los archivos enlazados
2. Fuente del sistema para Documentos públicos
3. Fuente del sistema para Mis documentos
4. Colección del sistema para Historial de uso
5. Colección inteligente
6. Grupos de colección inteligente
7. Grupo de colección
8. Colección
9. Fuente
10. Filtros de colección
11. Panel de colecciones
12. Extensor vertical de colecciones
13. Área de visualización
14. Número de elementos enumerados
15. Elemento seleccionado
16. Opciones de visualización
17. Opciones de clasificación
18. Cuadro de búsqueda
19. Editar botón de fuentes

Figura A.1 *eCatalogue*.

Un aspecto importante al trabajar con *eCatalogue* son los metadatos. Los metadatos de un elemento en el panel *eCatalogue* se pueden ver y editar utilizando el acceso directo *Ver/Editar metadatos* o haciendo clic derecho en un componente y seleccionando *ver metadatos*. Esto proporciona información

adicional sobre el componente, como su nombre, descripción, proveedor, autor, versiones, etc.(véase Figura A.2)

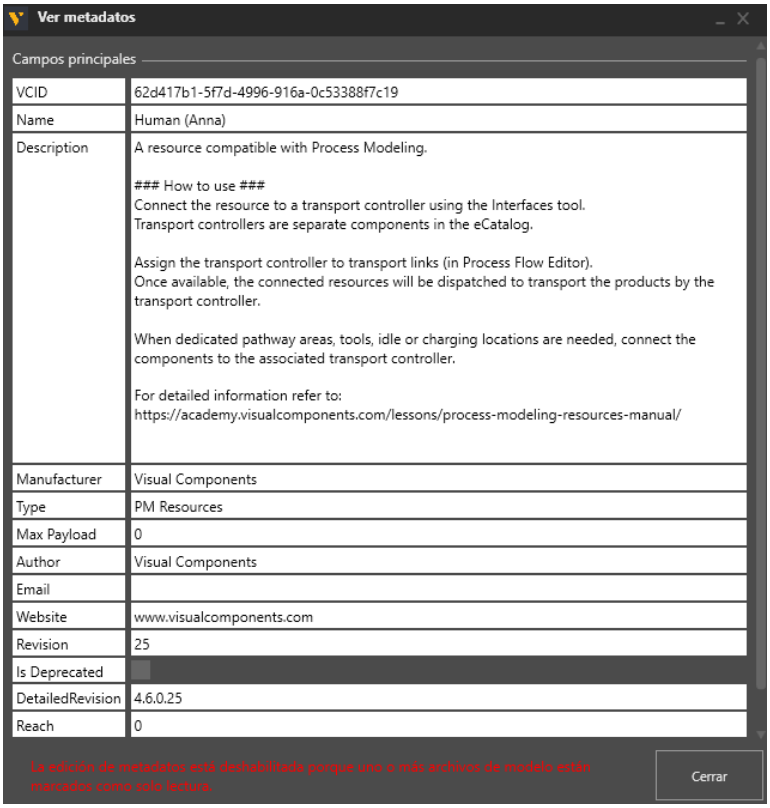


Figura A.2 Metadatos de un componente.

- *Procesos.* Como se mencionó en la sección 2.1.2, para realizar cualquier operación sobre procesos, es necesario acceder a la pestaña de proceso (ver Figura 2.3). Una vez allí, dependiendo de si se desea editar el flujo o el proceso, se seleccionará la sección correspondiente: *Flujo, Proceso* o *Producto.* En caso de editar el flujo, existen varios métodos disponibles:

1. La primera forma de trabajar con la herramienta de flujo es editando directamente en la sección del TFG, seleccionando un proceso origen y luego un proceso destino. Al hacerlo, aparecerá un icono triangular que permite definir qué agente, ya sea un robot o un humano, realizará el trabajo. Sin embargo, este método puede presentar errores.

2. La segunda forma de trabajar con la herramienta de flujo implica el uso del editor de flujo de proceso(véase Figura 3.2). Al hacer clic en el botón + ubicado en la parte derecha, se puede especificar el proceso origen, el proceso destino y el agente encargado de realizarlo. Además de ser más fiable, esta opción permite un mayor control sobre el procesamiento de los procesos, como activar la opción de que un nodo intermedio sea opcional o ejecutar varios nodos en paralelo en cualquier orden. Para acceder a estas opciones, es necesario hacer clic en el proceso correspondiente en la parte superior y, con el botón derecho del ratón, seleccionar la opción deseada.

   Para ver los procesos usados, se utiliza la sección *Procesos*, ubicada junto a la sección *Productos* (véase Figura 2.3). Al seleccionar uno de los procesos, se mostrarán tanto las variables definidas temporalmente en él como el editor de procesos (véase Figura 3.3).
   Para visualizar los productos y flujos de trabajo existentes, es necesario acceder a la sección *Productos*, ubicada en la parte superior derecha, donde se muestran tanto el flujo de trabajo como los objetos utilizados en su interior (véase Figura 2.3).

- *Código.* Como se ha comentado en el apartado 2.1.2, para el acceso al código de cualquier componente, se requiere ir a la pestaña de *Modelado* (véase Figura 2.4) y seleccionar la sección *Comportamiento* y dentro de esta, *ResourceScript*.

  Se ha delimitado claramente el código desarrollado por la autora de este TFG mediante comentarios como `TFG_CODE_INIT` y `TFG_CODE_EXIT`. Cualquier elemento que contenga la palabra TFG, ya sea en una señal o propiedad del

componente en el código, indica que ha sido creado o modificado en esa sección. Un caso especial es el componente de la estantería, cuyo comportamiento y señales se especifican en el código llamado *Buffer*.

Por otra parte, para cualquier archivo que se quiera almacenar en el equipo local definido en el código, si se cambia de equipo es necesario actualizar la ruta de acceso. Para localizar el código que especifica la ubicación del archivo, que se encuentra en el mismo archivo que el encargado de guardar los datos, es necesario acceder al controlador del robot y seguir los pasos indicados dichos en el apartado 2.1.2.

- *Simulación.* Es importante tener en cuenta el entorno en el que se reproduce el TFG, así como el tiempo que puede tardar y la posibilidad de detenerlo, pausarlo o repetirlo. También es relevante considerar el nivel de simulación, ya que el programa cuenta con una amplia capacidad gráfica y permite seleccionar diferentes tipos de simulación para acelerar el proceso. Hay 4 tipos distintos de nivel de simulación[7]:
    - Por defecto. La precisión la define el componente.
    - Detallado. Simula movimientos de los componentes lo más precisos posible, simulando por lo tanto todo el rango de movimiento del componente.
    - Equilibrado. Simula movimientos de los componentes de forma equilibrada con el rendimiento de la simulación, por lo que el componente se podría mover de punto a punto sin simular movimientos innecesarios de las articulaciones.

- Rápido. Simula movimientos de los componentes lo más rápidos posible, por lo que el componente se podría ajustar a la configuración de la articulación o saltar de punto a punto.

Además, se puede elegir entre el modo de tiempo real del Trabajo Fin de Grado que hace que la velocidad de la simulación se escala para que opere en tiempo real o el modo virtual que hace que la velocidad de la simulación dependa de la velocidad del ordenado, permitiendo una visualización más rápida.

Un punto importante de mencionar es el punto de reconfigurar (véase punto 2 de la Figura A.3). Al realizar esta acción, el diseño del entorno se restablece a su estado inicial. Esto implica que los componentes se reconfiguran a su estado inicial o guardado al comienzo de una simulación, y las variables del robot y las señales se restablecen a sus valores predeterminados. En otras palabras, el TFG se reinicia por completo a su estado previo a la simulación. Por desgracia, ha habido errores en este punto a la hora de tomar los resultados experimentales ya que no los volvía a su estado inicial, haciendo que la única solución sea pinchar varios clics al botón de reconfigurar para su buen funcionamiento.

1. Preferencias
2. Reconfigurar
3. Reproducir/Pausar
4. Reloj
5. Velocidad
6. Disminuir velocidad
7. Barra de velocidad
8. Incrementar velocidad
9. Grabar PDF 3D
10. Grabar vídeo
11. Grabar animación

Figura A.3 Controles de la simulación.

- *Panel de salida.* Proporciona información sobre eventos, comandos y otras acciones. Se puede usar para mostrar cómo están los valores tanto de las señales como de las variables, los posibles errores producidos al editar el código e incluso errores en la configuración del entorno creado.

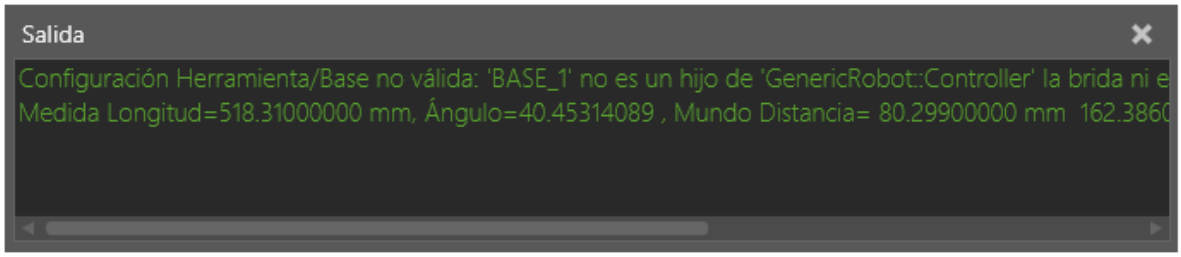


Figura A.4 Panel de salida.

- *Informe estadístico.* Dentro de la pestaña de inicio (véase Figura 2.2) se puede apreciar la sección de *Estadísticas*. Esta contiene los comandos más comúnmente usados para trabajar con estadísticas. Al entrar dentro, lo

primero que aparece son las opciones de diseño, que divide el contenido en pestañas separadas. Cada pestaña usa un diseño y sistema de cuadrícula para organizar los gráficos por columnas y filas.

Figura A.5 Elección del diseño estadístico.

Es importante mencionar el hecho de que si se cambia el diseño no se eliminan los gráficos ya creados a no ser que se cambie a un diseño con menos filas o columnas. Una vez elegido el diseño, se le da al botón *Crear* en el panel lateral (véase Figura A.5) y se elige un gráfico deseado(véase Figura A.6).

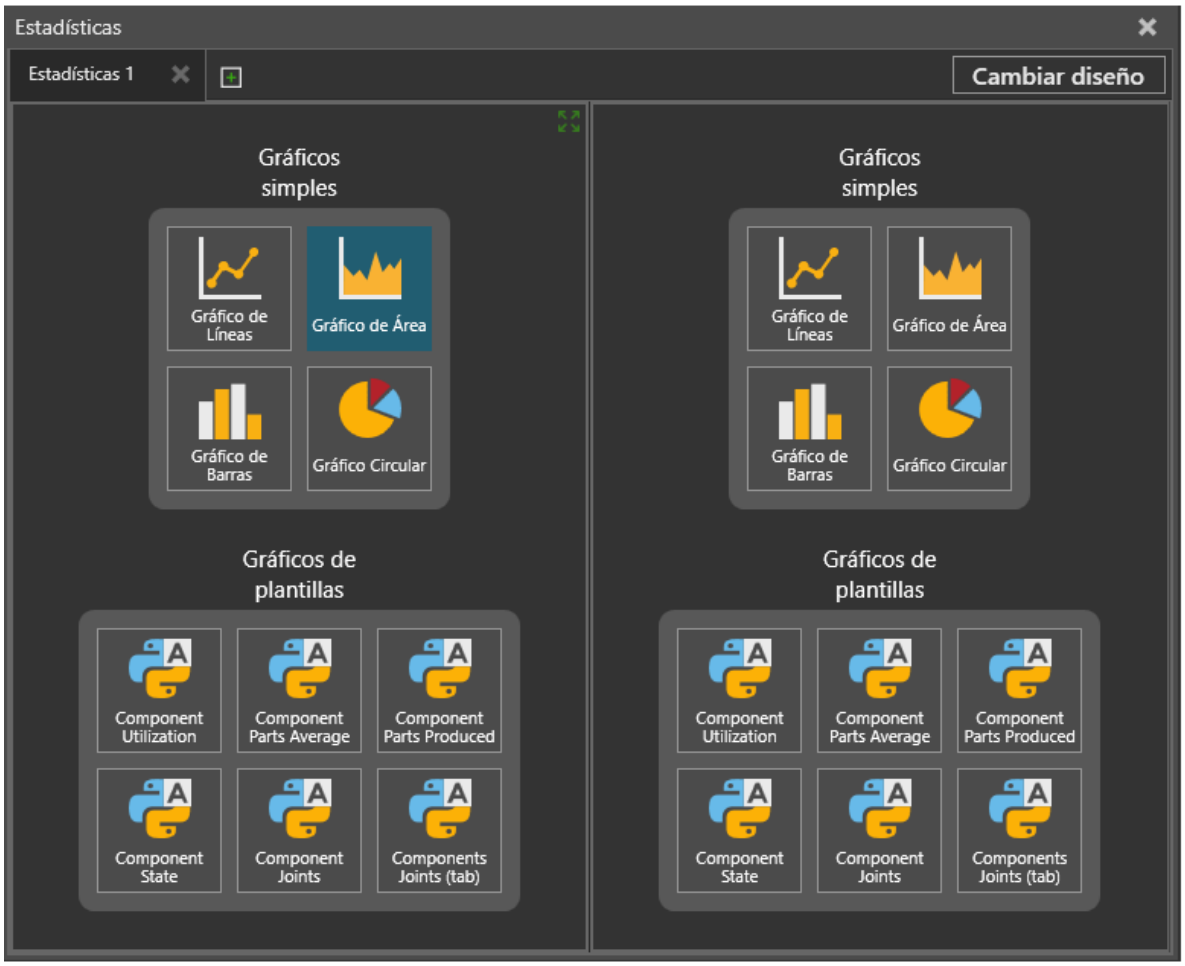


Figura A.6 Elección tipo de gráfico.

Elegido el gráfico (Figura A.6),para generarlo hay que tener en cuenta lo siguiente:

- Un gráfico traza una serie de datos o los valores de una propiedad de un componente.
- Una serie de datos puede contener datos de múltiples componentes que tengan una propiedad común. Un gráfico puede trazar más de una serie de datos y tener su propio intervalo de muestras.

- Un intervalo de muestras define la frecuencia con que se traza en un gráfico el valor de una serie de datos [7].
- Por defecto, todos los gráficos utilizan un intervalo de muestras global.

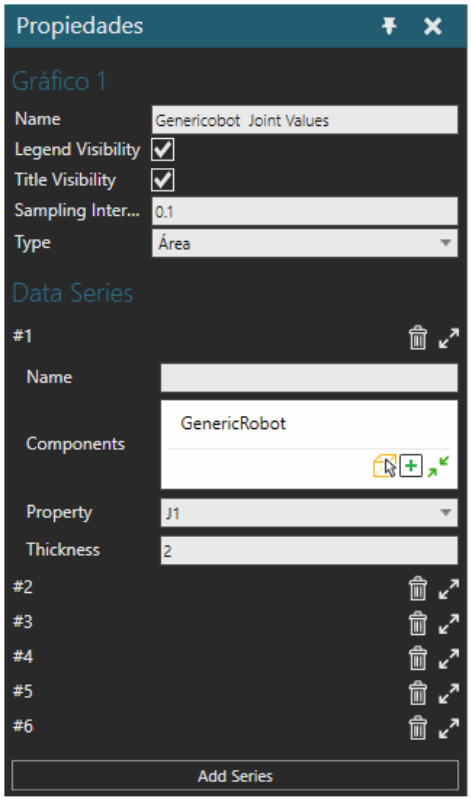


Figura A.7 Propiedad de una gráfica.

Como se aprecia en la Figura A.7, hay varios campos que se pueden proporcionar, como el nombre de la gráfica con la que se trabaja, el tipo de gráfica y la visibilidad de elementos primordiales en ésta.

Por otra parte, se define una serie de datos (en la parte baja el botón de añadir serie en la Figura A.7 junto con el orden de las series de datos en el gráfico) con varios valores seleccionables como su nombre, los componentes que lo definen y la propiedad de este que se quiere visualizar en el gráfico.

Por último, se incluye un botón para mejorar la visualización mediante el cambio del grosor de las series de datos en el gráfico.

Un gráfico trazará datos cuando se ejecute una simulación. Si se reconfigura y ejecuta una simulación nueva, se borrarán los datos de la simulación anterior

de todos los gráficos en el panel de control de estadísticas. Algo importante de mencionar es que un gráfico es un elemento de diseño, por lo que los gráficos se pueden guardar con un diseño. Esto no incluye los datos trazados. Es decir, cuando se guarda TFG se guardan la serie de datos que se han especificado pero no los resultados que han salido en la gráfica.

Para la exportación de datos, se pueden exportar los datos trazados en los gráficos a un archivo Excel o CSV. Para ello, situándose en la pestaña de inicio dentro de la sección de *Estadísticos*, haga clic en exportar gráfico y después uno de los siguientes:

- Para exportar datos a un archivo Excel, haga clic en Exportar a Excel. Los datos de cada gráfico se guardarán en su propia hoja en el archivo de Excel.
- Para exportar datos a un archivo CSV, haga clic en Exportar a CSV. Los datos de cada gráfico se guardarán en su propio archivo CSV.

Para cualquier otro tipo de operación o duda, consulte las opciones dadas en la pestaña de ayuda.

MATLAB

- *Entorno de trabajo.* Como ya se ha mencionado en el apartado 2.2.1, para trabajar con esta herramienta, hace falta conocer bien el escritorio que proporciona con sus elementos respectivos localizados (véase Figura 2.6). Es importante tener en cuenta la ruta en la que se esté trabajando debido a posibles errores de compilación a la hora de ejecutar tanto las instrucciones como los programas.

- *Código.* A la hora de trabajar en el código que se esté realizando, ya sea para la importación/exportación de datos o para obtener las gráficas, si se necesita crear una función, lo recomendable es hacerlo fuera del código principal ya que de esta manera se puede reutilizar la función cuantas veces se necesite e incluso exportarla.
- *Simulación.* Para ejecutar el código sobre el que se esté trabajando hay varias maneras:
    - Una opción más visual consiste en simplemente ir a la sección llamada *Run* y pulsar el botón llamado *Run* (véase Figura 2.6).

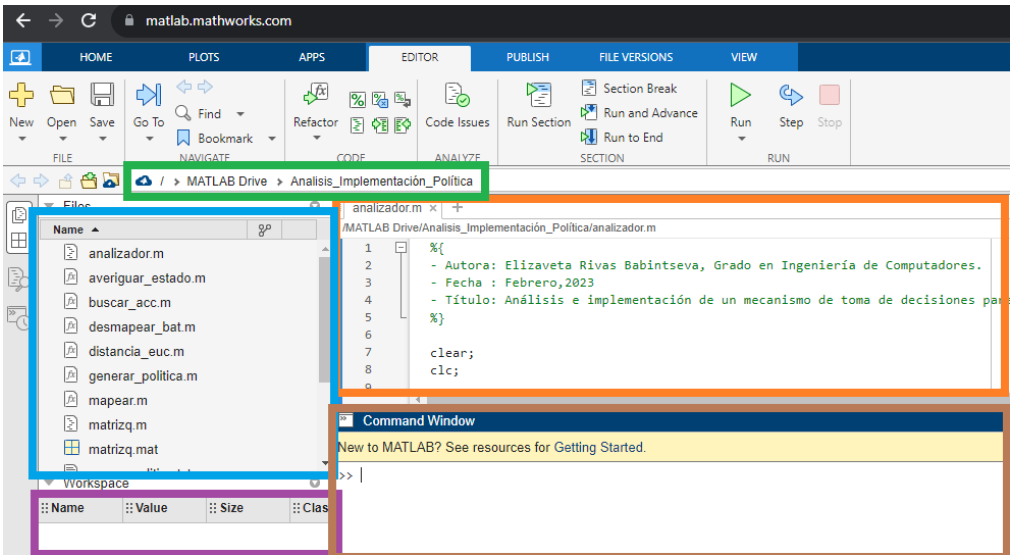


Figura 2.6 Interfaz de usuario del escritorio de *MATLAB*.

    - La segunda opción es seleccionar la parte del código que se quiere ejecutar y pulsar con el botón izquierdo del ratón en la opción *Evaluate Selection in Command Window,* o simplemente pulsar F9.
    - Por otra parte, hay una manera de depurar el código. Para ello, se requiere pinchar doble clic en la línea de código donde se quiere parar (aparece un rectángulo rojo como símbolo de que se ha parado) y pinchar al botón Run. Esto hará que cuando se ejecute el código, se

pare en la línea de código pedida haciendo que se pueda depurar el código línea por línea. Un ejemplo de ello está representado en la Figura A.8.

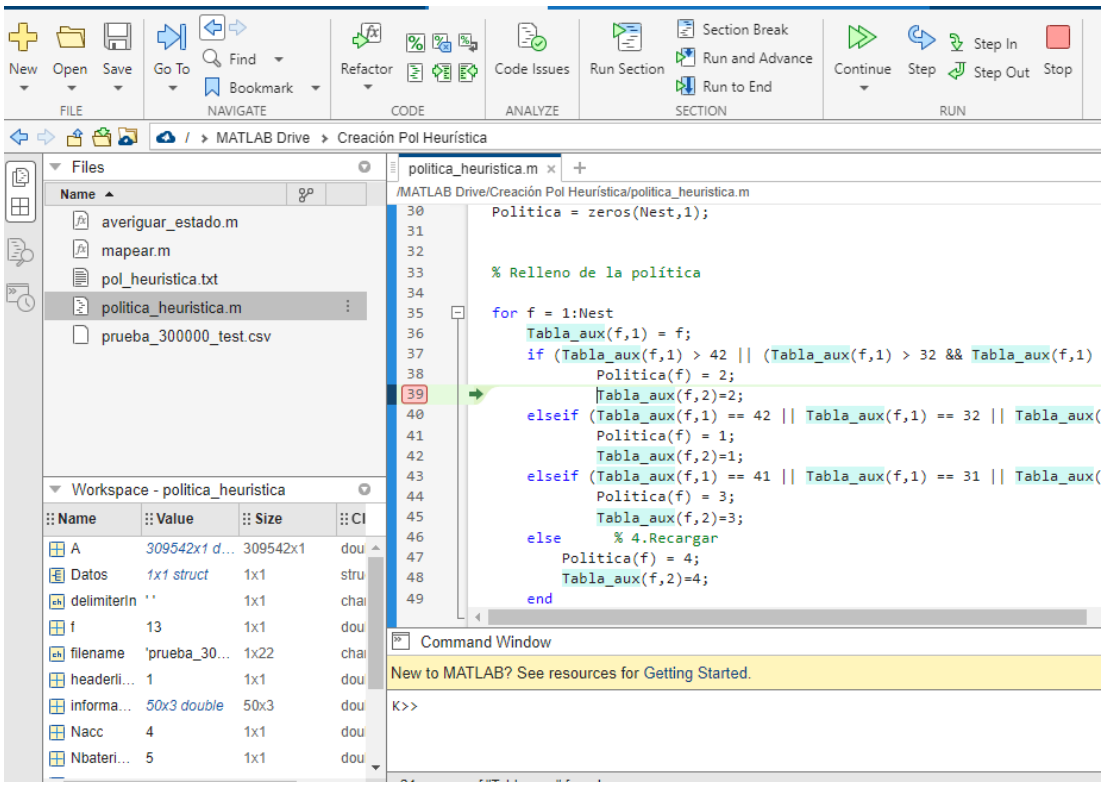


Figura A.8 Parada del código.

Referencias

[1] “3 claves para la convergencia IT - OT: Big Data, IoT y Cloud “, zemsaniaglobalgroup. https://zemsaniaglobalgroup.com/3-claves-para-la-convergencia-it-ot-big-data-iot-y-cloud/ (accedido el 8 de julio de 2023)

[2] “La importancia de la integración de T.I en fusiones y adquisiciones “, nedigital. https://www.nedigital.com/es/blog/la-importancia-de-la-integraci%C3%B3n-de-t.i-en-fusiones-y-adquisiciones (accedido el 8 de julio de 2023)

[3] “¿Qué es la Industria 4.0 y cómo funciona?”, IBM.com. https://www.ibm.com/es-es/topics/industry-4-0 (accedido el 6 de febrero de 2023)

[4] J. Zhang, P. Li, y L. Luo, “Digital twin-based smart manufacturing cell: Application Case, System Architecture and Implementation,” *Journal of Physics: Conference Series*, vol. 1884, no. 1, p. 012017, 2021 (accedido el 2 de noviembre de 2022)

[5] “MATLAB”, Mathworks.com. https://es.mathworks.com/products/matlab.html (accedido el 6 de febrero de 2023)

[6] Contributors to Wikimedia projects, “Markov decision process - Wikipedia”, en Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Markov_decision_process#Value_iteration (Accedido el 17 de febrero de 2023)

[7] “Visual Components - 3D manufacturing simulation software.”, Visual Components. https://www.visualcomponents.com/ (accedido el 21 de abril de 2023)

[8] “Programming application plugins with .NET Framework | Visual Components Academy”, Visual Components Academy. https://academy.visualcomponents.com/webinars/programming-application-plugins-with-net-framework/ (Accedido el 2 de agosto de 2023)

[9] Microsoft. *Visual Studio Code - Code Editing*. Accedido el 10 de diciembre de 2022. [En línea]. Disponible: https://code.visualstudio.com/

[10] “Introducing Visual Components 4.7 - Simplify the Complex - Visual Components”, Visual Components. https://www.visualcomponents.com/resources/blog/introducing-visual-components-4-7-simplify-the-complex/ (accedido el 9 de julio de 2023)

[11] “Simul8 | Fast, Intuitive Simulation Software for Desktop and Web”, Simul8 Simulation Software. https://www.simul8.com/ (accedido el 9 de julio de 2023)

[12] “Inicio FlexSim”, FlexSim. https://www.flexsim.com/es/ (accedido el 9 de julio de 2023)

[13] “Documentation- MATLAB & Simulink- MathWorks España”, MathWorks - Creadores de MATLAB y Simulink - MATLAB y Simulink - MATLAB & Simulink. https://es.mathworks.com/help/?s_tid=gn_supp (accedido el 9 de julio de 2023)

[14] “GNU Octave”, GNU Octave. https://octave.org/ (accedido el 9 de julio de 2023)

[15] “R: The R Project for Statistical Computing”, R: The R Project for Statistical Computing. https://www.r-project.org/ (accedido el 9 de julio de 2023)

[16] “Comparación de los cuatro principales lenguajes de programación de aprendizaje automático: R, Python, MATLAB, Octave - programador clic”, programador clic. https://programmerclick.com/article/64751476429/ (accedido el 9 de julio de 2023)

[17] R. S. Sutton y A. G. Barto, *Introduction to Reinforcement Learning*, 2a ed. London, England: The MIT Press, 2015, pp. 112-123 . Accedido el 24 de julio de 2023. [En línea]. Disponible: https://inst.eecs.berkeley.edu/~cs188/sp20/assets/files/SuttonBartoIPRLBook2ndEd.pdf

[18] “KPI: definición, para qué sirve y ejemplos prácticos”, Metricool. https://metricool.com/es/que-es-un-kpi/ (accedido el 20 de marzo de 2023)

[19] M. Á. R. Reina. “¿Por qué es la Mediana una medida muy importante? ¿Qué ventajas tiene respecto a la media? Definición”, econometria. https://www.eknowmetrics.com/single-post/2016/08/23/-por-qué-es-la-mediana-una-medida-muy-importante-qué-ventajas-tiene-respecto-a-la media#:~:text=Las%20ventajas%20con%20respecto%20a%20la%20media%20son:, es%20muy%20sensible%20a%20valores%20extremos)%20Más%20elementos (accedido el 14 de julio de 2023)

[20] “Visual Components”, Software UMA. https://software.uma.es/index.php/alumnos/visual-components (accedido el 6 de julio de 2023)

[21] “Servicio de túneles SSL”, Servicio central de informatica. https://www.uma.es/servicio-central-de-informatica/info/113153/tuneles-ssl (accedido el 6 de julio de 2023)

[22] “MATLAB”, Software UMA. https://software.uma.es/index.php/alumnos/matlab (accedido el 30 de enero de 2023)

[23] “Tutoriales de MATLAB y Simulink”, MathWorks - Creadores de MATLAB y Simulink - MATLAB y Simulink - MATLAB & Simulink. https://es.mathworks.com/support/learn-with-matlab-tutorials.html#:~:text=Aprenda%20los%20conceptos%20básicos%20de%20MATLAB%20®%20de,tiene%20acceso,%20incluidos%20los%20proporcionados%20por%20su%20universidad (accedido el 2 de febrero de 2023)

[24] “MATLAB Drive Files”, MATLAB Drive. https://drive.matlab.com/files/ (accedido el 14 de marzo de 2023)

[25] “Gráfica de líneas en 2D - MATLAB plot- MathWorks España”, MathWorks - Creadores de MATLAB y Simulink - MATLAB y Simulink - MATLAB & Simulink. https://es.mathworks.com/help/matlab/ref/plot.html (accedido el 20 de marzo de 2023)

[26] “Crear vectores, subindexar arreglos e iterar bucles for - MATLAB colon :- MathWorks España”, MathWorks - Creadores de MATLAB y Simulink - MATLAB y Simulink - MATLAB & Simulink. https://es.mathworks.com/help/matlab/ref/colon.html (accedido el 15 de marzo de 2023)

[27] “Matrices y arreglos- MATLAB & Simulink- MathWorks España”, MathWorks - Creadores de MATLAB y Simulink - MATLAB y Simulink - MATLAB & Simulink. https://es.mathworks.com/help/matlab/learn_matlab/matrices-and-arrays.html (accedido el 15 de marzo de 2023)

[28] Colaboradores de los proyectos Wikimedia, “Diseño asistido por computadora - Wikipedia, la enciclopedia libre”, Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Diseño_asistido_por_computadora (accedido el 3 de septiembre de 2023)

[29] “Supported CAD files for 3D Simulation”, Visual Components. https://www.visualcomponents.com/supported-cad-files/ (accedido el 3 de septiembre de 2023)

[30] Colaboradores de los proyectos Wikimedia, “Dinámica de sistemas - Wikipedia, la enciclopedia libre”, Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Dinámica_de_sistemas (accedido el 3 de septiembre de 2023)

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S. DE INGENIERÍA INFORMÁTICA

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga