

**UNIVERSIDAD MAYOR DE SAN SIMÓN**

**FACULTAD DE CIENCIAS Y TECNOLOGÍA**

**Corte de Piezas Rectangulares:Documentación Técnica**

**Docente:** Henrry Frank Villarroel Tapia

**Universitario(a):** Coca Quiroz Ever

**Carrera:** Ing. de Sistemas

**Asignatura:** Simulación de Sistemas

**Cochabamba – Bolivia**

ÍNDICE

Tabla de contenido

[1. Introducción 3](#_Toc194943467)

[2. Objetivo 3](#_Toc194943468)

[3. Marco Teórico 4](#_Toc194943469)

[4. Arquitectura del Proyecto 4](#_Toc194943470)

[5. Librerías Necesarias 4](#_Toc194943471)

[**Descripción de las librerías:** 5](#_Toc194943472)

[6. Implementación Técnica 5](#_Toc194943473)

[7. Pruebas de escritorio 6](#_Toc194943474)

[8. Conclusiones 7](#_Toc194943475)

## 1. Introducción

En el sector textil, uno de los desafíos operativos más recurrentes es la **gestión eficiente del material**, especialmente durante el proceso de **corte de piezas para confección**. La planificación inadecuada de este proceso puede conllevar a un elevado desperdicio de tela, incrementando los costos de producción y reduciendo la sostenibilidad del proceso. Por tanto, surge la necesidad de contar con **herramientas informáticas capaces de simular y optimizar la disposición de piezas sobre superficies textiles**, maximizando así el uso del material disponible.

Este proyecto propone una solución basada en **Python**, con énfasis en la simplicidad, automatización y posibilidad de extensión futura. Mediante técnicas de programación y simulación, se busca modelar el proceso de corte y evaluar diferentes configuraciones que permitan una mejor utilización del espacio disponible.

## 2. Objetivo

2.1 Objetivo General

Desarrollar un sistema de simulación computacional utilizando Python que permita **optimizar el posicionamiento de piezas rectangulares sobre una superficie textil**, reduciendo al mínimo el desperdicio de tela y mejorando la eficiencia del proceso de corte.

2.2 Objetivos Específicos

|  |  |
| --- | --- |
| Nº | Objetivo Específico |
| 1 | Diseñar un modelo de simulación para el problema del corte textil rectangular. |
| 2 | Implementar un algoritmo de posicionamiento secuencial eficiente en Python. |
| 3 | Utilizar herramientas como pandas y openpyxl para el procesamiento y exportación de datos. |
| 4 | Evaluar el desempeño del algoritmo con datos de prueba representativos. |
| 5 | Generar un reporte automatizado en Excel que sirva como apoyo en el proceso de toma de decisiones. |

Desarrollar un sistema en Python que permita simular y optimizar la disposición de piezas rectangulares sobre una plancha de tela, maximizando el uso del área y reduciendo el desperdicio. Este modelo debe ser capaz de:

* Identificar factores que influyen en la deserción de clientes.
* Predecir con una alta precisión qué clientes tienen mayor riesgo de abandonar la entidad.
* Permitir a la institución financiera tomar acciones preventivas para evitar la deserción.

## 3. Marco Teórico

**Problema del Corte (Cutting Stock Problem)**

El **Cutting Stock Problem (CSP)** es un problema clásico en la investigación de operaciones. Consiste en cortar piezas más pequeñas (rectangulares) a partir de objetos grandes (también rectangulares) de forma que se minimice el desperdicio.

**Algoritmos Heurísticos**

Se utilizan técnicas heurísticas porque permiten obtener soluciones razonablemente buenas en tiempo aceptable. En este caso se implementa una variación de la heurística **First Fit Decreasing (FFD)** adaptada a dos dimensiones.

## 4. Arquitectura del Proyecto

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | | Componente | Descripción | | main.py | Script principal que contiene todo el flujo del programa. Se ejecuta directamente y produce un archivo Excel como salida. | | pandas | Librería para estructuración y manipulación de datos tabulares. Se utiliza para organizar los resultados antes de exportarlos. | | openpyxl | Librería para exportar los resultados del DataFrame de pandas a un archivo .xlsx. | | entrada | Definida internamente como una lista de piezas, con nombre, ancho y alto. No requiere archivo externo. | | salida | Archivo Excel con la colocación final de las piezas, incluyendo coordenadas de ubicación. | |

## 5. Librerías Necesarias

Para la correcta ejecución del proyecto, es necesario instalar las siguientes librerías, las cuales permiten manipular datos, entrenar modelos y visualizar resultados:

Las librerías mencionadas anteriormente están en el repositorio digital de github en el siguiente link: <https://github.com/vetdy03/carteraDeClientesEnMicrofi.git> como tambien se adjuntará en un formato compreso en extensión .zip.

como también se adjunta el archivo requirements.txt el cual contiene el nombre de las librerías necesarias y una vez descargada en la misma ruta se debe proceder a ejecutar el siguiente comando

* pip install -r requirements.txt

Ó también ejecutarse el siguiente comando en la terminal o su consola.

* pip install pandas numpy matplotlib seaborn scikit-learn imbalanced-learn openpyxl

### **Descripción de las librerías:**

* **pandas**: Manipulación y análisis de datos en formato tabular.
* **numpy**: Soporte para operaciones matemáticas y manejo de arreglos.
* **matplotlib**: Creación de gráficos y visualizaciones.
* **seaborn**: Visualización de datos más avanzada y estilizada.
* **scikit-learn**: Implementación de modelos de Machine Learning y métricas de evaluación.
* **imbalanced-learn**: Métodos para balancear conjuntos de datos desbalanceados.
* **openpyxl**: Lectura y escritura de archivos Excel en Python.

## 6. Implementación Técnica

**Lenguaje de programación**

* **Python 3**, por su simplicidad, versatilidad y ecosistema de librerías científicas y de automatización.

**Estructura del código**

**1. Definición de datos**

*piezas = [*

*{"nombre": "A", "ancho": 40, "alto": 30},*

*{"nombre": "B", "ancho": 60, "alto": 50},*

*...*

*]*

**2. Ordenamiento por área descendente**

Para mejorar la eficiencia del uso del espacio, las piezas se ordenan por su área (ancho × alto) en orden descendente.

**3. Algoritmo de colocación**

La simulación posiciona las piezas horizontalmente en filas. Si una pieza no cabe en la fila actual, se crea una nueva fila debajo. Este enfoque asegura un llenado ordenado y rápido de la plancha.

**4. Validación de espacio**

Se valida que ninguna pieza sobrepase los límites definidos de la tela (por ejemplo, 100×100 cm).

**5. Exportación de resultados**

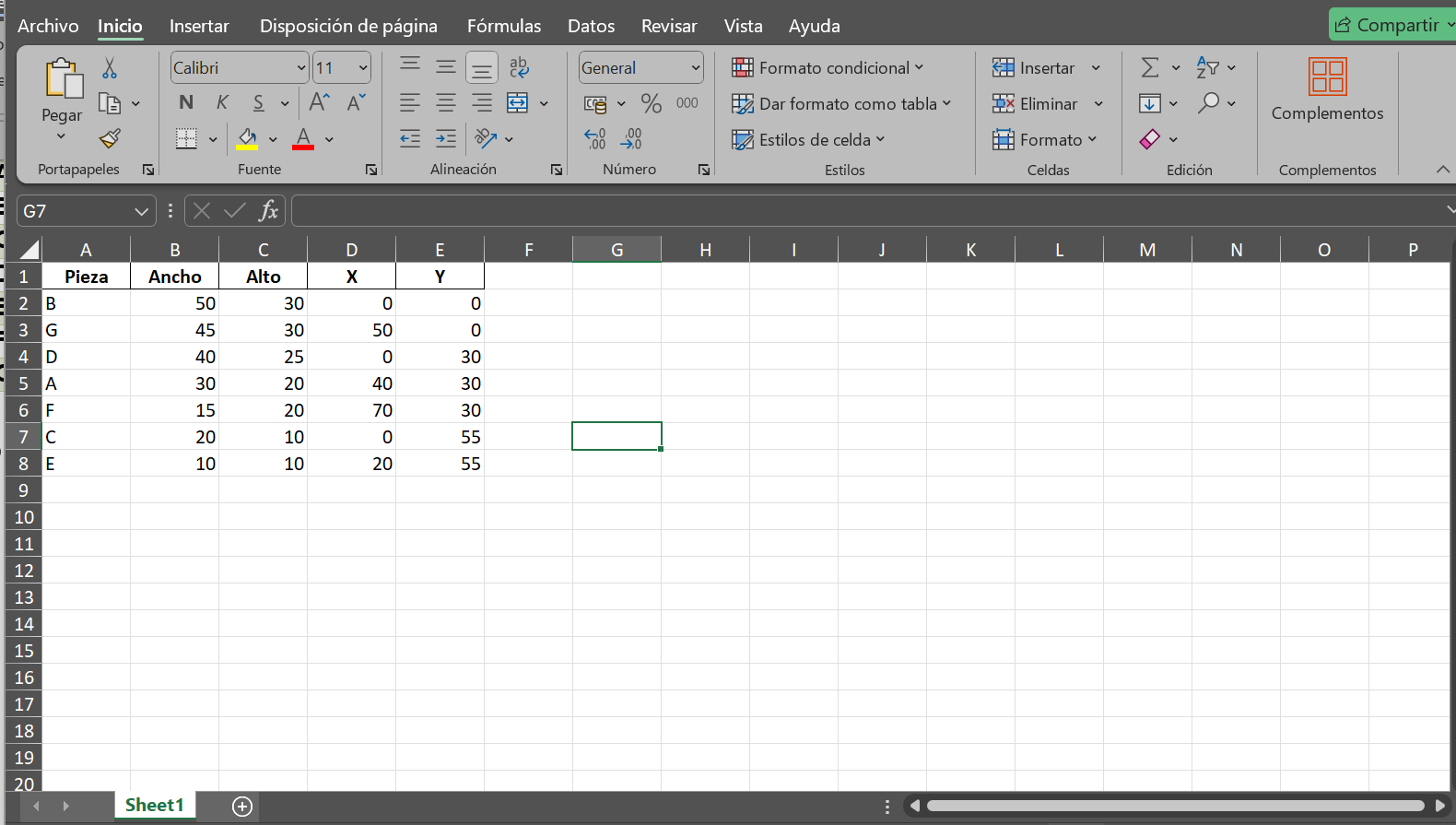
Los datos organizados se exportan a un archivo .xlsx mediante pandas y openpyxl, incluyendo:

* Nombre de pieza
* Ancho, alto
* Posición (x, y)
* Área ocupada

## 7. Pruebas de escritorio

|  |  |
| --- | --- |
| Prueba | Detalle |
| Datos pequeños | 6 piezas con dimensiones variadas, verificando posicionamiento correcto. |
| Datos extremos | Piezas grandes cercanas al tamaño del rollo de tela. Verificado rechazo por falta de espacio. |
| Prueba de bordes | Piezas que ajustan exactamente al ancho/alto disponibles. Verificación sin errores. |

|  |  |  |
| --- | --- | --- |
| Pieza | Ancho | Alto |
| A | 30 | 20 |
| B | 50 | 30 |
| C | 20 | 10 |
| D | 40 | 25 |
| E | 10 | 10 |
| F | 15 | 20 |
| G | 45 | 30 |



## Conclusiones

* El modelo implementado permite simular de manera efectiva y sencilla el proceso de corte sobre planchas textiles.
* Se logró una distribución ordenada y con mínimo desperdicio en casos básicos, ideal para microempresas o talleres.
* La herramienta es extensible: puede incluir rotación, forma irregular, múltiples planchas o conexión con interfaces gráficas.
* El uso de pandas y openpyxl garantiza una excelente compatibilidad para visualización y análisis externo en Excel.
* El sistema puede ser base para estudios más avanzados en simulación y optimización logística.