Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 2do. Semestre 2025

IMPLEMENTACIÓN DE SISTEMA DE OPTIMIZACIÓN PARA AGRICULTURA DE PRECISIÓN MEDIANTE AGRUPAMIENTO DE PATRONES DE FRECUENCIA

202300502 – Pablo Javier Alvarez Marroquin

Resumen

Este proyecto muestra una solución para optimizar el sistema de recolección de datos en sistemas de agricultura de precisión, mediante el agrupamiento de estaciones base que comparten patrones de frecuencia de transmisión. El Programa procesa archivos XML con información de campos agrícolas, estaciones base, sensores de suelo y sensores de cultivo, utilizando estructuras de datos abstractas implementadas en Python con la estructura de programación orientada a objetos. El sistema identifica patrones comunes en las matrices de frecuencia y agrupa estaciones con el mismo comportamiento, reduciendo así el número total de estaciones necesarias para transmitir datos en tiempo real hacia la nube. Esto optimiza el uso de recursos, disminuye costos operativos y mejora la eficiencia del sistema. Además, se generan un reporte XML y visualizaciones gráficas con Graphviz, facilitando la interpretación de resultados. La solución demuestra la aplicación de técnicas de agrupamiento y estructuras personalizadas para resolver problemas del mundo real en el ámbito agrícola.

Palabras clave
agricultura de precisión, optimización, patrones de frecuencia, XML, Graphviz.

Abstract

This project presents a solution to optimize the data collection system in precision agriculture by grouping base stations that share transmission frequency patterns. The program processes XML files containing information about agricultural fields, base stations, soil sensors, and crop sensors, using abstract data structures implemented in Python with an object-oriented programming structure. The system identifies common patterns in frequency matrices and groups stations with the same behavior, thus reducing the total number of stations required to transmit real-time data to the cloud. This optimizes resource usage, reduces operational costs, and improves system efficiency. In addition, an XML report and graphical visualizations with Graphviz are generated, facilitating the interpretation of results. The solution demonstrates the application of clustering techniques and custom data structures to solve real-world problems in the agricultural domain

Keywords
precision agriculture, optimization, frequency patterns, XML, Graphviz.

Introducción

La agricultura de precisión busca optimizar el uso de recursos agrícolas mediante la recolección y análisis de datos en tiempo real. Este proyecto ayuda a resolver el problema de reducir la cantidad de estaciones base necesarias para transmitir información de sensores de suelo y cultivo hacia una plataforma en la nube para almacenar la información. El proyecto propuesto utiliza programación orientada a objetos en Python y estructuras de datos abstractas para procesar archivos XML con la configuración de campos agrícolas. Con este programa que usa una estructura de agrupamiento de patrones de frecuencia, se logra identificar estaciones con comportamientos idénticos y consolidarlas, optimizando la infraestructura y reduciendo costos. La solución incluye la generación de reportes XML y visualizaciones gráficas con Graphviz, lo que permite comprender de forma clara la distribución y agrupamiento de estaciones. Este trabajo demuestra cómo técnicas computacionales pueden aplicarse eficazmente a problemas complejos en el sector agrícola.

Desarrollo del tema

a. Marco Teórico

- Fundamentos de agricultura de precisión
- técnicas de optimización
- Tipos de datos abstractos y estructuras enlazadas
- Procesamiento de XML en Python

b. Metodología Implementada

El sistema utiliza una matriz de frecuencias para identificar patrones comunes entre estaciones base, agrupando aquellas que comparten el mismo patrón de recepción de datos de sensores.

c. Implementación Técnica

Se desarrollaron estructuras de datos personalizadas (lista Enlazada) y clases para representar todos los componentes del sistema (estación, Sensor Suelo, Sensor Cultivo, Campo Agrícola).

d. Visualización de Resultados

El sistema genera reportes en XML y visualizaciones gráficas usando Graphviz para representar las relaciones entre estaciones y sensores.

Conclusiones

La creación y desarrollo del programa demuestra que el agrupamiento de patrones de frecuencia es una estrategia efectiva para optimizar sistemas de agricultura de precisión. El uso de TDA personalizados permitió un control detallado del procesamiento de datos, evitando estructuras nativas de Python. La solución es adaptable a diferentes configuraciones de campos agrícolas, y su integración con Graphviz facilita la interpretación visual de resultados. Este enfoque puede extenderse a otros dominios que requieran optimización de infraestructura basada en patrones de datos.

Referencias bibliográficas

Date, C. J. (2003). An Introduction to Database Systems. Addison-Wesley.

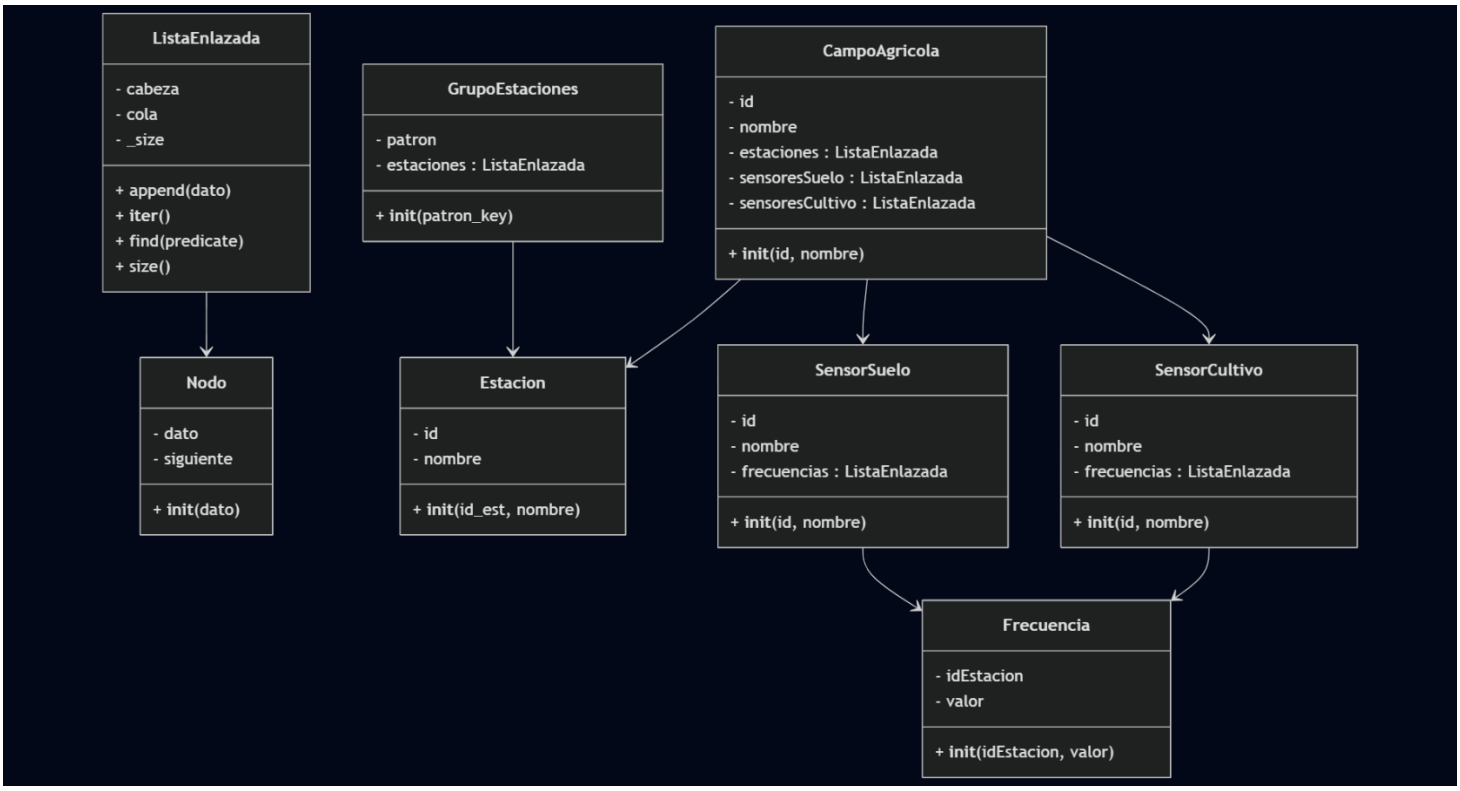
Graphviz.org.. (2023). Graphviz Documentation.

Python Software Foundation. (2023). Python XML Processing.

Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley.

C. J. Date, (1991). *An introduction to Database Systems.* Addison-Wesley Publishing Company, Inc.

Extensión:



```python
class Nodo:
    def __init__(self, dato):
        self.dato = dato
        self.siguiente = None


class ListaEnlazada:
    def __init__(self):
        self.cabeza = None
        self.cola = None
        self._size = 0

    def append(self, dato):
        nodo = Nodo(dato)
        if self.cabeza is None:
            self.cabeza = nodo
            self.cola = nodo
        else:
            self.cola.siguiente = nodo
            self.cola = nodo
        self._size += 1

    def __iter__(self):
        actual = self.cabeza
        while actual is not None:
            yield actual.dato
            actual = actual.siguiente
```

```python
class SensorSuelo:
    def __init__(self, id_s, nombre):
        self.id = id_s
        self.nombre = nombre
        self.frecuencias = ListaEnlazada()

class SensorCultivo:
    def __init__(self, id_t, nombre):
        self.id = id_t
        self.nombre = nombre
        self.frecuencias = ListaEnlazada()

class CampoAgricola:
    def __init__(self, id_c, nombre):
        self.id = id_c
        self.nombre = nombre
        self.estaciones = ListaEnlazada()
        self.sensoresSuelo = ListaEnlazada()
        self.sensoresCultivo = ListaEnlazada()

class GrupoEstaciones:
    def __init__(self, patron_key):
        self.patron = patron_key
        self.estaciones = ListaEnlazada()

def get_frecuencia_sensor(sensor, idEstacion):
    for freq in sensor.frecuencias:
        if freq.idEstacion == idEstacion:
            return freq.valor
    return 0
```

```python
def construir_patron_para_estacion(campo, idEstacion):
    suelo_parts = ""
    for sensor in campo.sensoresSuelo:
        val = get_frecuencia_sensor(sensor, idEstacion)
        if suelo_parts == "":
            suelo_parts = str(val)
        else:
            suelo_parts = suelo_parts + "," + str(val)
    cultivo_parts = ""
    for sensor in campo.sensoresCultivo:
        val = get_frecuencia_sensor(sensor, idEstacion)
        if cultivo_parts == "":
            cultivo_parts = str(val)
        else:
            cultivo_parts = cultivo_parts + "," + str(val)
    return suelo_parts + "|" + cultivo_parts

def agrupar_estaciones_por_patron(campo):
    grupos = ListaEnlazada()
    for estacion in campo.estaciones:
        key = construir_patron_para_estacion(campo, estacion.id)
        grupo = grupos.find(lambda g: g.patron == key)
        if grupo is None:
            nuevo = GrupoEstaciones(key)
            nuevo.estaciones.append(estacion)
            grupos.append(nuevo)
        else:
            grupo.estaciones.append(estacion)
    return grupos
```