



MANUAL PARA LA OPTIMIZACIÓN DE LA LOCALIZACIÓN DE SENSORES DE DETECCIÓN TEMPRANA DE INCENDIOS FORESTALES

PRE-REQUISITOS

Antes de empezar con los pasos requeridos para optimizar la localización de sensores de detección temprana de incendios forestales es importante que el usuario cumpla con las siguientes condiciones:

- Tener instalado Anaconda.
- Tener instalado Julia.
- Tener instalado Gurobi y contar con licencia vigente.
- Tener instalado Python.
- Tener imagen de vista aérea del área a la que se quiera localizar los sensores, la imagen debe estar en formato .png.

OPTIMIZACIÓN DE LA LOCALIZACIÓN DE SENSORES DE DETECCIÓN TEMPRANA DE INCENDIOS FORESTALES

PASO 1: DATOS PRELIMINARES

- a. Tener la imagen aérea del área en la que se van a localizar los sensores guardada en el dispositivo en el que se va a correr el modelo. La imagen debe ser guardada con el nombre “1” y en formato **.png**.
- b. Abre el archivo “ANEXO 14” en Jupyter Notebook

```
jupyter Anexo # . Generador mapa de bits (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

In [ ]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import cv2

In [ ]: imgcv2.imread('1.png',0)
Dfaspd.DataFrame(a)
Dfa.to_csv("img1.csv")

In [ ]: dfaspd.read_csv("img1.csv",sepa=";")

In [ ]: dfaspd.round(df/np.max(df),2)

In [ ]: df = df.drop('Unnamed: 0', 1)
mat=df.to_numpy()

In [ ]: tamanos=100
x=[]
y=[]
c=[]
for i in range(tamanos):
    for j in range(tamanos):
        x.append(i)
        y.append(j)
        c.append(mat[i,j])
len(x)

In [ ]: fig,ax=plt.subplots(figsize=(20,20))
plt.scatter(x,y,c,c,c,markers="s")

In [ ]:
```

- c. Ejecuta el código y ubica el archivo img1.csv en tu dispositivo

PASO 2: EJECUCIÓN MODELO

- a. Abre el archivo “ANEXO 13” en Jupyter Notebook
- b. Ve a la línea de código 8 y establece los siguientes datos de entrada: Número de escenarios a simular, radio de un sensor, tamaño de la cuadrícula y número de sensores.

```
Ejecuta el modelo de simulación que determina las ubicaciones óptima en una serie de escenarios
generados mediante Montecarlo, si se quieren gráficos de los escenarios cambiar el false por
true

In [ ]: num_escenarios=100 #
radio=3
tamanos=50
sensores=3

quemado=Initialize(tamano)
df = DataFrame(CSV.File("img3.csv"))
weights=Matrix(df)

#Estamos cogiendo solo un pedacito de la imagen, la imagen completa es muy muy grande para el modelo de optimización
weights=weights[1:tamano,1:tamano]
weights=round.(weights/maximum(weights);digits=3)

valores=[]
for i in 1:tamano, j in 1:tamano
    push!(valores,weights[i,j])
end

# as
Veg_arbol=weights.>percentile(valores, 20)
P=[]

for i in 1:num_escenarios
    Matriz_prioridades=CreateEscenario(Veg_arbol,zeros(tamano,tamano),weights,tamano, false)
    push!(P,Matriz_prioridades)
end

MC, l_sa=Matrix_Cubiertos(tamano,radio)
locaciones=Create_model(P,MC, sensores, l_s, tamano)
```

- c. Si se desean gráficos de los escenarios cambia el *false* , en la línea 27, por *true*.

- d. Ejecuta las primeras 8 líneas del código.
- e. En la última línea de la celda de *Output* se encuentra el resultado de las ubicaciones de los sensores.

PASO 3: VALIDACIÓN DE MÉTRICAS

- a. Ejecuta la línea 9 del código.
- b. Establece el número de repeticiones para la simulación.

```
In [ ]: n_repeticiones=100

cont=0
tiempo_det=[]
area_burnt=[]
while cont<n_repeticiones
    coverage, tiempos=RunSimulationWide(Veg_arbol,zeros(tamano,tamano),weights,locaciones,radio,tamano)
    if length(coverage)>0
        println("tiempo hasta detección: ", tiempos[1], " ---- ", "% Área quemada: ", coverage[1]*100)
        push!(tiempo_det,tiempos[1])
        push!(area_burnt,coverage[1]*100)
        cont+=1
    end
end

println("IC tiempo detección: ", t_interval(tiempo_det))
println("IC Área quemada : ", t_interval(area_burnt))
```

- c. Ejecuta la línea 10 del código.
- d. En la celda de *Output* se encuentran las métricas Tiempo hasta detección y Porcentaje de área quemada hasta detección por cada iteración.