Asignación óptima de móviles ante saltos de alarma mediante algoritmos genéticos

Diego Araujo

14 de junio de 2025

1. Introducción

La asignación eficiente de recursos en situaciones críticas constituye un desafío habitual en la operativa diaria de empresas que gestionan servicios de respuesta ante eventos. En particular, el despliegue óptimo de móviles de respuesta ante eventos simultáneos, como los saltos de alarma en sistemas de seguridad, representa un desafío logístico relevante para empresas del rubro. Este trabajo se centra en una versión estática del problema, donde un conjunto de alarmas se activan de forma simultánea en una ciudad, y deben ser atendidas por una flota limitada de móviles que parten desde una misma base.

El objetivo es minimizar el tiempo de respuesta por los móviles al atender las alarmas, contemplando además aspectos como el equilibrio en la carga de trabajo entre unidades y el orden de atención dentro de cada ruta. Para ello, se implementó una solución basada en algoritmos genéticos, una técnica inspirada en la evolución natural que ha demostrado ser efectiva en problemas de optimización combinatoria complejos.

Este documento describe la formulación del problema, los detalles de implementación del algoritmo genético, los resultados obtenidos, la visualización de soluciones y una discusión sobre las limitaciones y posibles mejoras futuras.

2. Descripción del problema

Se consideró un escenario urbano, geográficamente acotado al departamento de Montevideo, en el cual se simularon 40 puntos georreferenciados representando alarmas activadas (figura 1). Estos puntos fueron generados aleatoriamente dentro de un polígono correspondiente a los límites de la ciudad, garantizando una ubicación válida en zonas terrestres.

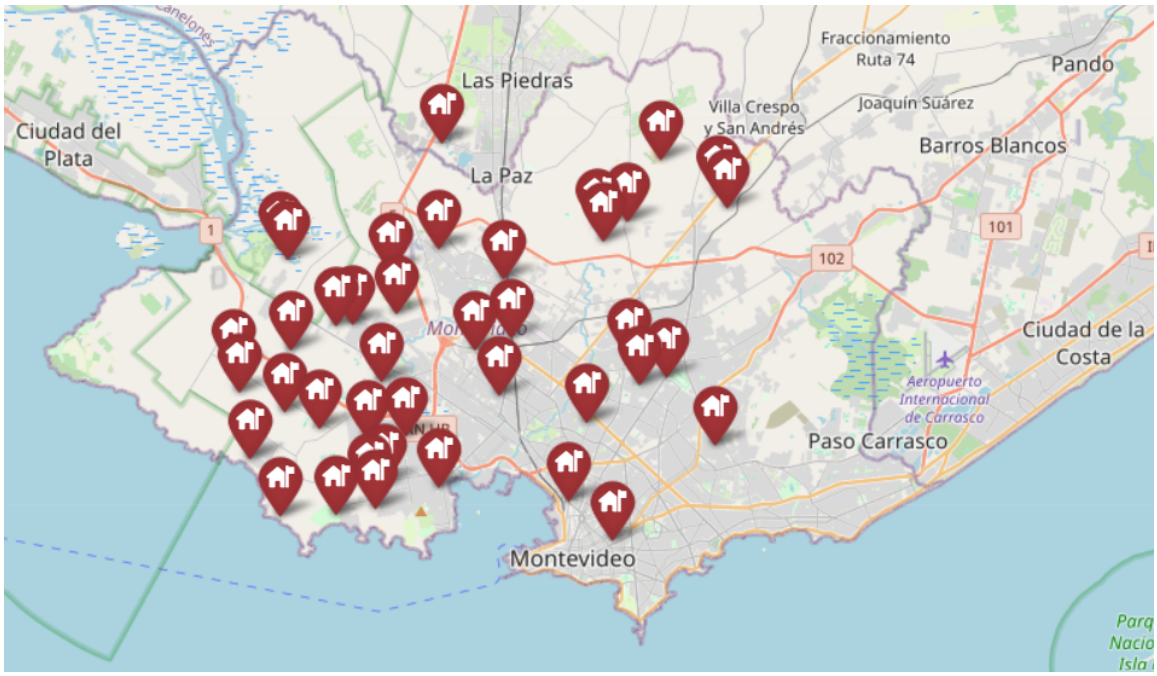

Figura 1: Simulación de puntos que representan alarmas activadas.

Los móviles de respuesta, en este caso 3 unidades, parten desde una misma base fija ubicada también dentro del polígono. No existen restricciones de capacidad ni prioridades diferenciadas entre las alarmas, y todos los eventos deben ser atendidos por un único móvil. Un punto clave del problema es que la planificación es estática: se asume que todas las alarmas se activan al mismo tiempo (un caso común de falsas alarmas en situaciones de condiciones climáticas adversas), sin aparición de nuevos eventos durante la ejecución.

La métrica de evaluación principal es la sumatoria del tiempo de respuesta de los tres móviles, considerando el trayecto desde la base hasta cada alarma asignada en orden. Para esta estimación se tiene en cuenta lo siguiente:

- Se toma la distancia euclidiana para calcular diferencias de distancias entre puntos.
- Las distancias del punto anterior se convierten a kms suponiendo que $1° \approx 111 kms$. [1]
- La distancia en kilómetros finalmente se convierte en tiempo dada la velocidad máxima fijada para los móviles.

Claramente, el utilizar la distancia euclidiana entre coordenadas geográficas implica una simplificación del problema al no considerar aspectos reales como el tráfico, direcciones de calles o tiempos de viaje. Este problema se podría solventar, haciendo uso de alguna API (OSM, Google Maps) quizás, aunque escapa al alcance de este trabajo.

3. Algoritmo genético: representación y operadores

Para abordar la resolución, se diseñó un algoritmo genético que opera sobre una representación compuesta: cada individuo de la población consiste en una lista de tres rutas

[1] https://es.wikipedia.org/wiki/Latitud

(una por móvil), donde cada ruta es una secuencia ordenada de índices que indican qué alarmas atiende ese móvil y en qué orden. Esta codificación garantiza que cada alarma esté presente una sola vez en toda la solución.

El proceso evolutivo comienza con una generación inicial construida de forma aleatoria, y luego, en cada iteración (generación), se aplican los siguientes operadores:

- **Evaluación de *fitness***: se calcula la suma de las distancias recorridas por los tres móviles según las rutas propuestas y conversión de estas distancias en tiempo. También se incorpora una penalización por desbalance de carga entre móviles, a través de la desviación estándar del número de puntos asignados.
- **Selección**: se emplea selección por torneo, donde se eligen los mejores individuos entre subconjuntos aleatorios de la población.
- **Cruza (crossover)**: se intercambian segmentos de rutas entre dos padres, cuidando de mantener la cobertura completa y sin duplicados.
- **Mutación**: se realizan perturbaciones locales, como el intercambio de dos puntos dentro de una ruta o el traspaso de una alarma de un móvil a otro.
- **Elitismo**: se conserva el mejor individuo de cada generación para asegurar convergencia.

Estos puntos están contemplados en el siguiente código:

```python
historial_fitness = []

def algoritmo_genetico(N_GENERACIONES = N_GENERACIONES,
                       TAM_POBLACION = TAM_POBLACION,
                        TASA_MUTACION = TASA_MUTACION):
    poblacion = crear_poblacion(TAM_POBLACION, N_ALARMAS)
    #print(poblacion)
    mejor_solucion = None
    mejor_fitness = float('inf') # Para poder minimizar, se arranca con un valor "grande"
    for gen in range(N_GENERACIONES):
        fitnesses = [calculo_fitness(ind, distancias) for ind in poblacion]
        historial_fitness.append(min(fitnesses))
        for ind, fit in zip(poblacion, fitnesses):
            if fit < mejor_fitness:
                mejor_fitness = fit
                mejor_solucion = ind[:]
        nueva_poblacion = []
        while len(nueva_poblacion) < TAM_POBLACION:
            padre1 = seleccion_torneo(poblacion, fitnesses)
            padre2 = seleccion_torneo(poblacion, fitnesses)
            hijo = crossover_ox(padre1, padre2)
            if random.random() < TASA_MUTACION:
                hijo = mutacion_swap(hijo)
            nueva_poblacion.append(hijo)
        poblacion = nueva_poblacion
        if gen % 20 == 0:
            print(f"Generación {gen} \t  ---- \t Mejor fitness: {mejor_fitness:.2f}")
    return mejor_solucion, mejor_fitness

mejor_solucion, mejor_fitness = algoritmo_genetico()
```

Figura 2: Código principal del algoritmo genético.

El algoritmo se ejecuta por un número predefinido de generaciones (en principio 1000), y se registra el mejor individuo obtenido según la función de *fitness*. Las funciones auxiliares que conforman dicho código (y su explicación), se pueden ver con mayor detalle en el notebook .

En cuanto al cálculo del *fitness*, se puede observar en el siguiente código:

```python
def calculo_fitness(cromosoma, distancias,
                    balancear_carga=BALANCEAR_CARGA,
                    penalizacion_carga=PENALIZACION_CARGA,
                    usar_tiempo=True,
                    velocidad_kmh=VELOCIDAD_KMH):
    rutas = dividir_ruta(cromosoma)
    totales = []
    for ruta in rutas:
        if len(ruta) == 0:
            totales.append(0)
            continue
        total = 0
        anterior = 0
        for punto in ruta:
            d = distancias[anterior][punto]
            total += d
            anterior = punto
        totales.append(total)

    distancia_total = sum(totales)
    if balancear_carga:
        desviacion = np.std(totales)
        distancia_total += penalizacion_carga * desviacion
    if usar_tiempo:
        # Este cálculo está basado en la conversión de grados a km, suponiendo que 1° ~ 111 kms
        distancia_total = (distancia_total * 111) / velocidad_kmh * 60
    return distancia_total
```

Figura 3: Código del cálculo de la función de *fitness*.

4. Visualización de resultados geoespaciales

Una de las fortalezas del trabajo es la integración de mapas interactivos utilizando la librería *Folium*, que permite inspeccionar visualmente las rutas obtenidas. Se representaron las siguientes capas:

- **Puntos de alarma**: marcadores con íconos `house-flag` en rojo, numerados según el orden de atención.
- **Base de partida**: marcador azul fijo.
- **Rutas de cada móvil**: líneas de diferentes colores que conectan los puntos asignados a cada móvil.

En este caso, luego de la ejecución del algoritmo genético con parámetros iniciales fijados, el resultado de las rutas óptima es el que se observa en la figura 4.

Figura 4: Rutas optimizadas a través del algoritmo genético.

Esta visualización permite verificar fácilmente la asignación correcta de alarmas, la coherencia espacial de cada ruta, y detectar posibles ineficiencias o zonas mal cubiertas.

5. Resultados obtenidos

Los resultados (de esta ejeución inicial) muestran una asignación eficiente con rutas compactas para cada móvil. La figura 5 presenta un gráfico que permite observar cómo evoluciona la calidad de las soluciones generadas por el algoritmo genético a lo largo de las generaciones.

Figura 5: Evolución de la función de *fitness* para cada generación.

El eje X indica el número de generación, mientras que el eje Y representa el valor mínimo de *fitness* observado en esa generación. En este contexto, el *fitness* corresponde al tiempo total estimado en minutos que los móviles necesitan para atender todas las

alarmas asignadas.

El análisis de la curva muestra una fase inicial con mejoras rápidas (exploración del espacio de soluciones), seguida por una zona de estabilización donde el algoritmo refina soluciones cada vez más cercanas al óptimo. A partir de cierto punto (alrededor de la generación 400), las mejoras se vuelven marginales, lo que parece indicar cierta convergencia del proceso evolutivo.

En la figura 6, se observa que en la mejor solución obtenida, aunque la distribución no es perfectamente equitativa, los tiempos están en un rango bastante acotado (entre 53 y 61 minutos), lo que sugiere que el algoritmo logró mantener una distribución razonable sin comprometer la eficiencia global. De hecho, y desde una perspectiva operativa, que ningún móvil vaya más allá de la hora de recorrido sugiere que la solución es viable y eficiente, especialmente si se asume que este recorrido se realiza en una única salida o ronda.

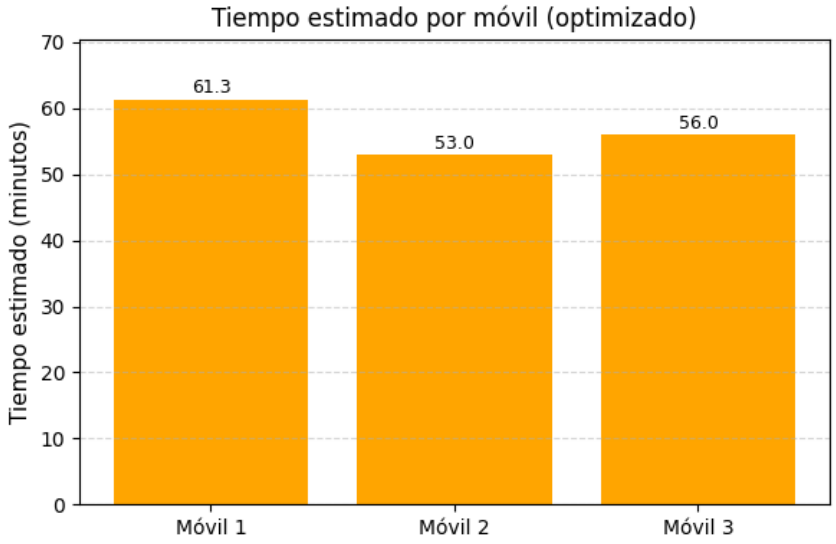

Figura 6: Tiempos totales de recorrida por móvil.

También se realizaron pruebas de comparación entre una solución aleatoria y la solución lograda mediante algoritmos genéticos, teniendo los siguientes resultados:

- **Solución aleatoria:** 617.94 minutos.
- **Solución AG:** 173.64 minutos.

Esto representa una mejora superior al 70 % en eficiencia operativa. El resultado evidencia que una asignación arbitraria puede ser extremadamente ineficiente, generando recorridos largos y desbalance entre móviles. En contraste, el algoritmo genético encuentra una estructura de rutas mucho más eficiente, logrando una distribución espacial adecuada y menor tiempo de atención global.

6. Búsqueda de hiperparámetros con Optuna

El desempeño del algoritmo genético está influenciado por sus parámetros, como la tasa de cruzamiento, mutación, tamaño de población y el uso de elitismo. Para optimizar estos valores, se integró la biblioteca *Optuna*, una herramienta de optimización bayesiana que permite explorar el espacio de hiperparámetros de manera eficiente.

Se definió una función objetivo que ejecuta el algoritmo genético con un conjunto de parámetros propuestos y devuelve el tiempo total obtenido por la mejor solución. Optuna realizó múltiples pruebas (trials) variando parámetros como:

- Tasa de cruzamiento (entre 0.5 y 0.9)
- Tasa de mutación (entre 0.1 y 0.5)
- Tamaño de población (entre 50 y 200)
- Cantidad de generaciones (entre 100 y 500)
- Penalización por desbalance de carga (entre 0 y 5)
- Velocidad de los móviles (entre 30 y 50)

De esta manera se logró llegar a valores optimizados de hiperparámetros para utilizar en el modelo, que devolvió los siguientes resultados.

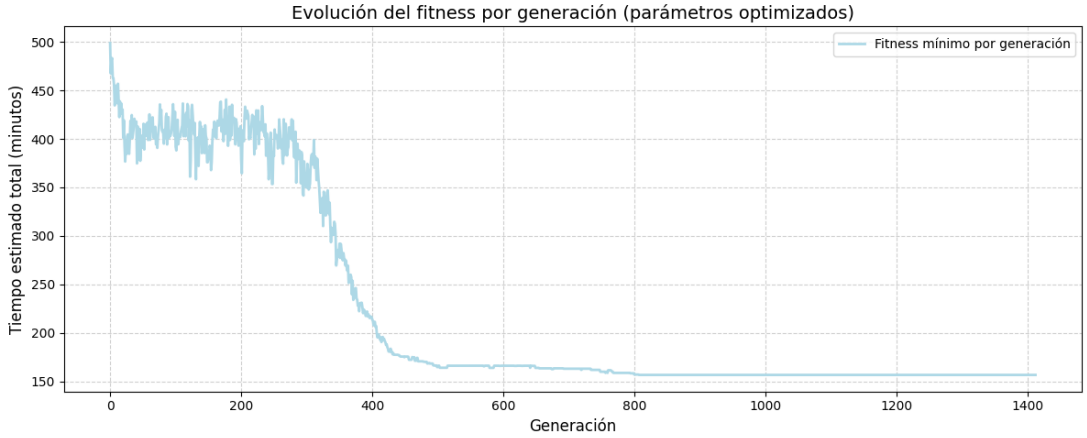

Figura 7: Evolución de la función de *fitness* en modelo optimizado con Optuna.

La figura 7, así como se veía anteriormente en el modelo sin optimizar, muestra una fase inicial de exploración con alta variabilidad en los valores de *fitness*, seguida por una etapa de mejora sostenida a partir de la generación 400, donde se reduce significativamente el tiempo estimado total. A partir de la generación 600, el modelo alcanza una clara estabilización, con valores cercanos a 158 minutos, lo que indica convergencia hacia una solución eficiente.

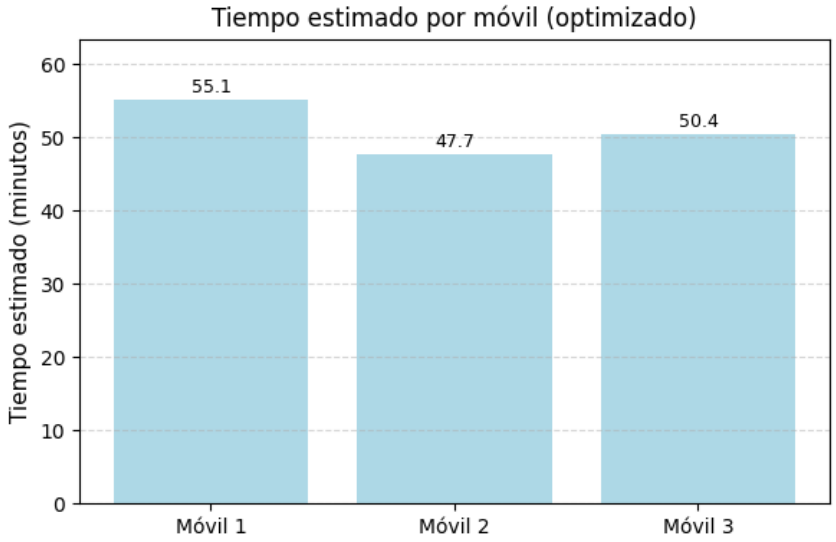

Figura 8: Tiempos totales por móvil en modelo optimizado con Optuna.

En la figura 8, se observa un valor total que ronda los 150 minutos aprox., lo que es muy bajo en comparación con:

- El *fitness* de una solución aleatoria, que rondaba los 618 minutos.
- El mejor *fitness* alcanzado sin optimización de hiperparámetros, que estaba entre 170–180 minutos, según lo mostrado en gráficos anteriores.

Esto implica una mejora sustancial adicional gracias a la optimización de parámetros, por lo que confirma que Optuna fue capaz de encontrar una configuración más eficiente que la usada previamente, incluso cuando el algoritmo ya era competitivo. Las nuevas rutas, luego de esta optimización, se pueden ver en la figura 9.

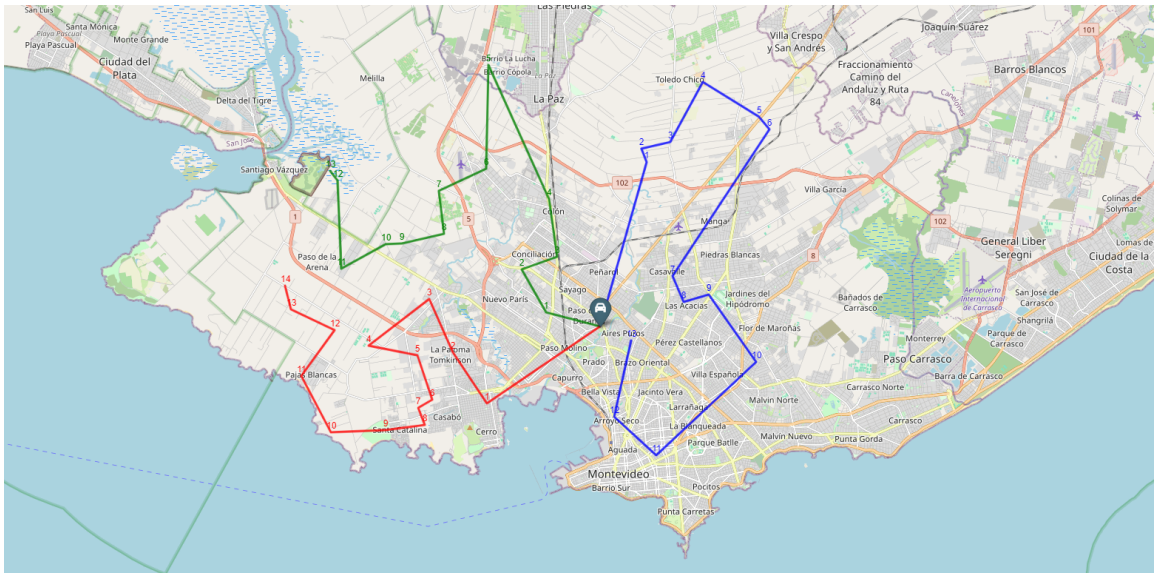

Figura 9: Nuevas rutas luego de optimización del modelo mediante Optuna.

7. Limitaciones del enfoque y mejoras futuras

Aunque los resultados fueron alentadores, existen limitaciones importantes:

- **El modelo fue simplificado**: la distancia euclidiana no refleja la distancia efectiva de desplazamiento, si bien puede dar una buena aproximación.
- **Existe falta de tiempos reales**: no se consideran duraciones efectivas de desplazamiento ni tiempos de atención.
- **No hay prioridades en las alarmas**: todas las alarmas se tratan como iguales, lo que no es realista en sistemas de seguridad. Por lo general, existen prioridades tanto en eventos como en clientes.
- **La planificación fue estática**: no se contempla la aparición dinámica de nuevos eventos durante las recorridas.

Estas limitaciones y supuestos se podrían levantar a través de:

- Integrar APIs como *Nominatim* de OSM o *Google Maps* para obtener distancias reales y tiempos estimados.
- Incorporar prioridades por tipo de cliente o urgencias.
- Ampliar el modelo a uno dinámico o semi-dinámico, donde las rutas se actualicen al recibir nuevos eventos.
- Aplicar restricciones operativas, como tiempo máximo de atención o zonas de cobertura, además de agregar tiempos en cada instalación.

8. Conclusiones

Este trabajo demuestra que los algoritmos genéticos, correctamente diseñados e integrados con herramientas de visualización y optimización de hiperparámetros, pueden ser una solución efectiva para la planificación operativa de servicios de respuesta ante alarmas.

A pesar de la simplificación del modelo, se obtuvo una mejora sustancial en la eficiencia de rutas, con un enfoque replicable y adaptable a contextos reales mediante ajustes adicionales.

A su vez, la posibilidad de visualizar los resultados sobre mapas interactivos refuerza su utilidad para la toma de decisiones operativas, y sugiere que esta línea de trabajo puede escalarse para aplicaciones reales en empresas de seguridad, logística o atención de emergencias.