

Guía de Estructuras de Datos del Sistema de Evacuación

Estructuras de Entrada (Inputs)

Tu política de evacuación recibirá tres parámetros principales:

1. city: CityGraph

Representa el diseño de la ciudad como un grafo de NetworkX.

```
class CityGraph:
    graph: networkx.Graph          # Grafo de la ciudad
    starting_node: int             # Nodo inicial
    extraction_nodes: List[int]    # Puntos de extracción posibles
```

Ejemplo:

```
city = CityGraph()
# El grafo contiene:
# - Nodos: Representados por IDs (números enteros)
# - Aristas: Conexiones entre nodos con atributo 'weight' (distancia)
print(city.starting_node) # Ej: 0
print(city.extraction_nodes) # Ej: [12, 25, 31]
```

2. proxy_data: ProxyData

Contiene las lecturas ambientales para nodos y aristas.

PROF

Indicadores de Nodo (proxy_data.node_data[node_id])

Cada nodo tiene los siguientes indicadores (valores de 0 a 1):

- **radiation_readings**: Niveles de radiación
- **thermal_readings**: Firmas de calor (indica presencia de zombies)
- **seismic_activity**: Inestabilidad estructural
- **signal_strength**: Calidad de comunicaciones
- **population_density**: Niveles de actividad
- **emergency_calls**: Señales de socorro
- **structural_integrity**: Condición del edificio

Ejemplo:

```
# Datos para un nodo específico
node_data = proxy_data.node_data[5]
print(node_data) # Ejemplo de salida:
{
    'radiation_readings': 0.75,      # Alta radiación
    'thermal_readings': 0.82,      # Alta presencia de zombies
    'seismic_activity': 0.45,      # Inestabilidad moderada
    'signal_strength': 0.23,      # Mala comunicación
    'population_density': 0.67,    # Actividad significativa
    'emergency_calls': 0.91,      # Muchas llamadas de emergencia
    'structural_integrity': 0.34    # Estructura débil
}
```

Indicadores de Arista (proxy_data.edge_data[(node1,node2)])

Cada arista tiene los siguientes indicadores (valores de 0 a 1):

- **structural_damage**: Bloqueo de ruta
- **signal_interference**: Interrupción de comunicaciones
- **movement_sightings**: Detección de actividad
- **debris_density**: Niveles de obstáculos
- **hazard_gradient**: Cambios ambientales

Ejemplo:

```
# Datos para una arista específica
edge_data = proxy_data.edge_data[(1,2)]
print(edge_data) # Ejemplo de salida:
{
    'structural_damage': 0.65,      # Ruta bastante bloqueada
    'signal_interference': 0.43,    # Interferencia moderada
    'movement_sightings': 0.78,    # Alta actividad
    'debris_density': 0.56,        # Obstáculos moderados
    'hazard_gradient': 0.34        # Cambios ambientales moderados
}
```

PROF

3. max_resources: int

Número máximo total de recursos que se pueden asignar.

Estructura de Salida (Output): PolicyResult

Tu política debe retornar un PolicyResult que contiene:

```
class PolicyResult:
    path: List[int]          # Secuencia de nodos a visitar
```

```
resources: Dict[str, int] # Asignación de recursos
```

Ejemplo de Salida:

```
# Ejemplo de un PolicyResult válido
return PolicyResult(
    path=[0, 4, 7, 12], # Ruta desde starting_node hasta un
    extraction_node
    resources={
        'explosives': 3, # Para despejar rutas bloqueadas
        'ammo': 4, # Para encuentros con zombies
        'radiation_suits': 3 # Para zonas radiactivas
    } # Total de recursos <= max_resources
)
```

Evaluación de la Solución

El evaluador verificará:

1. Validez de la ruta:

- Comienza en starting_node
- Termina en un extraction_node
- Nodos consecutivos están conectados en el grafo

2. Uso de recursos:

- No excede max_resources
- Suficientes para superar obstáculos:
 - radiation_suits: Necesarios cuando radiation_readings > 0.35
 - ammo: Necesario cuando thermal_readings > 0.45
 - explosives: Necesarios cuando structural_damage > 0.4

3. Éxito de la misión:

- Llega a punto de extracción
- Mantiene recursos suficientes durante todo el recorrido
- Completa la misión en tiempo razonable

Ejemplo de Eventos Durante la Simulación

```
# Ejemplo de registro de eventos generado
"""
Eventos de la Misión:

Paso 1:
- Llegada al nodo 0
```

- Movimiento al nodo 4

Paso 2:

- Llegada al nodo 4
- Alta radiación detectada (nivel: 0.78)
- Uso de traje de radiación
- Movimiento al nodo 7

Paso 3:

- Llegada al nodo 7
- Horda de zombies encontrada (nivel: 0.82)
- Uso de munición contra zombies
- Ruta bloqueada al nodo 12 (nivel de daño: 0.65)
- Uso de explosivos para despejar ruta
- Movimiento al nodo 12

Paso 4:

- Llegada al nodo 12
- Punto de extracción alcanzado exitosamente en el nodo 12

Estado Final de Recursos:

- Trajes de Radiación: 2 restantes
- Munición: 3 restantes
- Explosivos: 2 restantes

""""

Consejos para el Desarrollo

1. Análisis de Datos:

- Utiliza los indicadores de nodo para identificar zonas peligrosas
- Usa los indicadores de arista para evaluar la dificultad de las rutas
- Considera la correlación entre diferentes indicadores

2. Planificación de Ruta:

- Implementa algoritmos de búsqueda de ruta (ej: Dijkstra, A*)
- Considera pesos personalizados basados en los indicadores
- Ten en cuenta la disponibilidad de recursos al planificar

3. Gestión de Recursos:

- Estima las necesidades de recursos basándote en los indicadores
- Mantén un margen de seguridad para imprevistos
- Considera la longitud total de la ruta al asignar recursos

4. Pruebas:

- Usa `run_simulation.py` para pruebas individuales detalladas
- Utiliza `run_bulk_simulations.py` para pruebas masivas
- Analiza los registros de eventos para identificar puntos de fallo

Datos Generados

Los datos de simulación se almacenan en el directorio `data/policies/EvacuationPolicy/experiments/`. Cada experimento genera la siguiente estructura:

```
data/policies/EvacuationPolicy/experiments/exp_<timestamp>_<uuid>/
├── summary.json           # Resumen del experimento
├── cities/               # Datos por ciudad
│   └── city_<id>/
│       ├── definition.json # Definición de la ciudad
│       ├── proxy_data.json # Datos ambientales
│       └── mission_results.json # Resultados de la misión
└── visualizations/       # Gráficos y análisis
```

summary.json

Contiene la metadata y resultados agregados del experimento:

```
{
  "experiment_id": "exp_20240315_a1b2c3d4",
  "timestamp": "2024-03-15T10:30:00",
  "configuration": {
    "type": "single_run",
    "n_nodes": 30,
    "seed": 42
  },
  "results": {
    "n_simulations": 10,
    "success_rate": 0.75,
    "avg_path_length": 12.3,
    "avg_time": 15.7,
    "resource_usage": {
      "avg_allocated": 8.5,
      "avg_used": 6.2,
      "avg_needed": 7.0,
      "efficiency": 0.85
    }
  }
}
```

PROF

cities/<city_id>/definition.json

Define la estructura de la ciudad:

```
{
  "metadata": {
```

```

        "timestamp": "2024-03-15T10:30:00",
        "n_nodes": 30,
        "max_resources": 10
    },
    "graph": {
        "nodes": [...], # Lista de nodos con posiciones
        "edges": [...]  # Lista de conexiones con pesos
    },
    "configuration": {
        "start_node": 0,
        "extraction_nodes": [12, 25]
    }
}

```

cities/<city_id>/proxy_data.json

Contiene las lecturas ambientales:

```

{
    "metadata": {
        "timestamp": "2024-03-15T10:30:00"
    },
    "indicators": {
        "nodes": {
            "0": {
                "radiation_readings": 0.75,
                "thermal_readings": 0.82,
                # ... otros indicadores
            },
            # ... otros nodos
        },
        "edges": {
            "(0,1)": {
                "structural_damage": 0.65,
                "movement_sightings": 0.78,
                # ... otros indicadores
            },
            # ... otras aristas
        }
    }
}

```

cities/<city_id>/mission_results.json

Resultados detallados de la misión:

```

{
    "success": true,

```

```

"path_taken": [0, 4, 7, 12],
"time_taken": 15.7,
"resources": {
    "allocated": {
        "explosives": 3,
        "ammo": 4,
        "radiation_suits": 3
    },
    "used": {
        "explosives": 2,
        "ammo": 3,
        "radiation_suits": 2
    }
},
"events": [
    {
        "step": 1,
        "type": "move",
        "description": "Llegada al nodo 0"
    },
    # ... más eventos
]
}

```

Visualizaciones

El directorio [visualizations/](#) contiene varios tipos de gráficos:

- Tasas de éxito por tamaño de ciudad
- Análisis de eficiencia de recursos
- Correlaciones entre indicadores ambientales
- Patrones de uso de recursos
- Relaciones tiempo-distancia
- Mapas de calor de peligros

PROF

Estos datos son útiles para:

1. Analizar el rendimiento de tu política
2. Identificar patrones de fallo
3. Optimizar la asignación de recursos
4. Entender la relación entre indicadores ambientales y éxito de la misión

Visualizaciones Generadas

Visualizaciones de Ejecución Individual (run_simulation.py)

Actualmente solo muestra en pantalla:

- Grafo de la ciudad con:
 - Nodos coloreados según nivel de peligro

- Ruta planificada resaltada
- Puntos de inicio y extracción marcados
- Registro de eventos en tiempo real

Visualizaciones de Ejecuciones Masivas (run_bulk_simulations.py)


Se guardan en

`data/policies/EvacuationPolicy/experiments/exp_<timestamp>/visualizations/:`

1. success_rates.png

Tasas de Éxito por Tamaño de Ciudad


- |— Eje X: Número de nodos
- |— Eje Y: Tasa de éxito (%)

 success_rates

2. resource_efficiency.png

Eficiencia de Uso de Recursos

- |— Eje X: Tipo de recurso
- |— Eje Y: Cantidad
- |— Barras: Asignado vs Usado

 resource_efficiency

3. proxy_correlations.png

Correlaciones de Indicadores Ambientales


- |— Matriz de correlación
- |— Intensidad de color indica fuerza de correlación

 proxy_correlations

4. time_distance.png

Relación Tiempo-Distancia

- |— Eje X: Longitud de ruta
- |— Eje Y: Tiempo de misión
- |— Puntos coloreados por éxito/fallo

 time_distance

5. resource_impact.png

Impacto de Recursos en Éxito


- Eje X: Recursos utilizados
- Eje Y: Tasa de éxito
- Líneas por tipo de recurso

 resource_impact

6. hazard_heatmap.png

Mapa de Calor de Peligros

- Intensidad de color por nivel de peligro
- Superposición de diferentes tipos de amenaza

 hazard_heatmap


Visualizaciones por Ciudad

Para cada ciudad en [cities/<city_id>/visualizations/](#):

1. city_graph.png

Grafo Detallado de la Ciudad

- Nodos: Coloreados por tipo de peligro
- Aristas: Grosor indica dificultad
- Ruta: Resaltada en color
- Recursos: Iconos donde se utilizaron

 city_graph

2. resource_usage.png

Uso de Recursos en la Ciudad

- Eje X: Paso de la misión
- Eje Y: Recursos restantes

 resource_usage

Interpretación de Visualizaciones

1. Análisis de Éxito

- Las tasas de éxito por tamaño ayudan a entender la escalabilidad
- Patrones de fallo revelan limitaciones de la política

2. Uso de Recursos

- La eficiencia muestra si hay sobre/sub-asignación
- El impacto ayuda a priorizar recursos

3. Patrones Ambientales

- Correlaciones revelan relaciones entre peligros
- Mapas de calor muestran zonas de alto riesgo

4. Rendimiento Temporal

- Relación tiempo-distancia identifica rutas ineficientes
- Puntos atípicos señalan situaciones problemáticas

Nota: Las imágenes de ejemplo mostradas arriba son representativas. Las visualizaciones reales variarán según tus datos específicos.