دانشگاه بوعلی سینا

Algorithm design course
Final project

Star Wars Project report

Prof. Mirhossein Dezfoulian

Amir Mohammad Fazeli 40212358030

Pouya Tavakoli 40212358011

Introduction

This project focuses on a strategic missile routing and targeting algorithm designed to stimulate decision-making in presence of adversarial surveillance (spies). The algorithm is structured to evaluate all possible attack paths from multiple base cities to multiple target cities, assess risks based on surveillance, and execute missile strikes to maximize inflicted damage while minimizing exposure risk.
The implementation is built using the Boost Graph Library in c++, and relies on object-oriented abstractions.

Graph Construction

- An **undirected graph** is used where each **vertex** represents a city and each **edge** represents a possible direct route between cities
- Edges are created **only between city pairs whose euclidean distance is less than or equal to uncontrolled distance**. This ensures realism by limiting missile range.
- Cities can be:
  - **Base Cities**: Launch missiles.
  - **Target Cities**: Receive missile attacks.
  - **Normal Cities**: Intermediate locations for path routing.

Scenario 1

1. Pathfinding Algorithm

The algorithm uses **Breadth-First Search (BFS)** to explore all possible paths between **base cities** and **target cities** through the constructed graph.

key steps

1. **Identify Base and Target Vertices**:
   - Loop through all vertices and classify them based on their type.
2. **BFS Traversal**:
   - Start BFS to find **all simple paths** to target cities.
   - Keep track of visited paths.
   - Record the cities passed through.
   - Count the number of cities with spies.

3. **Store as PathInfo:**Each path is represented by a `PathInfo` struct containing:
    - `base`: Start city (vertex descriptor)
    - `target`: End city (vertex descriptor)
    - `cities`: Names of cities in the path
    - `spyCount`: Number of cities with spies along the path

4. **Path Sorting**
   Paths from each base city are stored in a map and sorted in ascending order of spyCount, so the safest paths are tried first during attacks.

2. Attack Phases

The attack phase is divided into two main stages:

1. **Phase 1: Safe Attacks**
   This phase attempts to:
    - Use the **safest available path** (least spies) from each base.
    - Fire missiles **only if their stealth rating exceeds the number of spies** in the path.
    - Calculate damage.

   If no missile can safely pass (stealth < spy count), that base is added to a **fallback pool**.
2. **Phase 2: Fallback Attacks**
   in this stage:
    - Bases that couldn't attack safely in Phase 1 coordinate to perform a **joint attack**.
    - The algorithm:
        1. Evaluates all remaining missiles from fallback bases.
        2. Identifies the **best target** by estimating:
            - Total missiles directed at the target.
            - The target's **defense level**.
            - **Bypassed missiles** = `max(0, totalMissiles - defense)`.
            - Total potential **damage** = `bypassed × missile damage`.
        3. The **target with maximum bypassed damage** is selected.
        4. launched missiles first blocked by defense number, and remaining missiles inflict damage.

3. Damage Calculation & Reporting

1. Damage Tracking
   - Damage dealt by each base is accumulated into a global total.
   - Blocked missiles are also tracked and reported.
2. Summary Output
   The algorithm prints:

   - Paths used for attacks.
   - Damage per base and per target.
   - Number of blocked vs bypassed missiles.
   - Total final damage.

Complexity Analysis:

- Initialization O(V^2)
- Path Finding O(V*(V+E))
- Attack O(Missiles × Targets)

Scenario 2

In this scenario, the goal is to attack enemy cities with A type missiles.
as the uncontrolled distance for all A type missiles is 500 the cities that their distance is less than 500 are connected in *initialize* function.

Algorithm Workflow

1. **Initialization**
   - **Input**:
     Uncontrolled distance threshold (500 km)
   - **Logic**:
     Calculate pairwise distances between all cities
     Connect cities within threshold distance
     Build undirected graph for pathfinding
   - **Output**:
     Connected city network ready for traversal
2. **Path Finding (BFS)**
   Process:
   Identify all base and target vertices For each base:
   - Initialize BFS queue with base vertex
   - Explore all neighboring cities level-by-level
   - Record paths terminating at targets

   Store path metadata:
     - City sequence
     - Total distance
     - Number of spy cities encountered

   Complexity: O(V+E) per base
3. **Path Organization**
   - Group paths by originating base
   - Prioritizes paths with:
     - Fewest spy cities (stealth)
     - Shortest distance

4. **Attack Simulation**
   - **Safe Attacks (Phase 1)**
     in this phase we only fire missile that are not revealed by spies so there is guaranteed hit and damage for all missiles fired in this phase.

     For each base:

     - Check missile inventory
     - Select best path (lowest spy count + distance)
     - Launch if safe and possible (bestPath.spyCount < missile.getStealth() && missile.getOveralDistance() >= bestPath.distance)

   - **Fallback Attacks (Phase 2)**
     now it's time to fire revealed missiles, in this phase is activated when no more safe paths available. We try to maximize damage by concentrating firepower on the weakest target. We call this “rapid fire”.
     - Group exposed missiles by target city
     - Calculate defense penetration:
     - Execute "Rapid-Fire" strategy: Concentrate firepower on weakest target.

5. **Complexity Analysis:**

   - Initialization $O(V^2)$
   - Path Finding $O(V*(V+E))$
   - Attack $O(\text{Missiles} \times \text{Targets})$

Scenario 3

This scenario focuses on distributing the missiles across the bases so that it inflects maximum damage.
This scenario works slightly differently with the previous two. First of all we set uncontrolled distance to maximum uncontrolled distance among the available missiles so that every route less than that is connected.

Algorithm Workflow

1. **Initialization**
   - **Input:**
     Maximum uncontrolled distance among the available missiles(900km).
   - **Logic:**
     Calculate pairwise distances between all cities
     Connect cities within threshold distance
   - **Output:**
     Connected city network ready for traversal
2. **Path Finding (BFS)**
   Process:
   - Identify each base and target.
   - Initialize BFS queue with base vertex.
   - Explore all neighboring cities level-by-level.
   - Record paths terminating at targets

   Store path metadata:
   - Cities sequence
   - Spies encountered
   - Total distance traversed
   - Maximum gap between cities
3. **Construct Paths**
   using the stored metadata we construct path for each missile:
   - Compare maximum gap with missiles uncontrolled distance
   - Compare total distance with missiles total distance
   - categorize missiles into two groups, **safe** and**, revealed** based on spies encountered

4. **Attack**
   our goal here is to maximize damage to do so:

- Prioritize missiles so that missiles with most damage are launched in a safe path.
- If our base is out of capacity we simply delete it.
- If not we shoot the missiles through a safe path and update base capacity.

if no safe path is found, we use fallback attacks:

- Collect **fallback bases** by checking the revealed paths and their capacity.
- While any missile is left or capacity, each time it gathers targets, does an attack and chooses the maximum damage dealt.

5. **Complexity Analysis:**
   - Initialization $O(V^2)$
   - Path Finding $O(V*(V+E))$
   - Attack $O(\text{Missiles} \times \text{Targets})$

Scenario 4

In this scenario, the goal is to attack targets using given missiles for each city. missiles types are A, B and C so the uncontrolled distance is set to 900 which is max of all.

Algorithm Workflow

1. **Initialization**
    - **Input:**
      Maximum uncontrolled distance among the available missiles(900km).
    - **Logic**:
      Calculate pairwise distances between all cities
      Connect cities within threshold distance
      Build undirected graph for pathfinding
    - **Output**:
      Connected city network ready for traversal.
2. **Path Finding**
    - **Process:**
        - Identify all base and target vertices from the graph.
        - For each base city, perform Breadth-First Search (BFS) to explore all reachable cities.
        - Record all paths terminating at target cities.
        - For each path, compute:
            - Total distance (sum of edge weights).
            - Spy count (number of cities with spies along the path).
            - Max gap (longest gap between consecutive cities).
    - **Complexity**: O(V*(V+E)) per base (BFS for each base vertex).
3. **Path Organization**
    Categorize paths by missile type, feasibility and safety.
    - **Feasibility**: first check feasibility by checking total distance and comparing max gap with uncontrolled distance.
    - **safety:** A path is **"safe"** if spyCount < missile_stealth. Otherwise, it is **"revealed"**.

4. **Attack Simulation**
    - **Phase 1: Safe Attacks**
      For each base and missile type:
        - Retrieve the best "safe" path (prioritizing low spy count and distance).
        - Launch all available missiles of that type via the safe path.

- **Phase 2: Fallback Attacks (Rapid-Fire)**
    - **Trigger**: When no safe paths remain for exposed missiles.
    - **Strategy**
        - For each base with exposed missiles, identify all reachable targets.
        - For each target, compute:
            - **Total missiles** available to attack it.
            - **Total damage** potential.
        - Select the target with the highest **bypassed damage**.
        -
    - **Execution:**
        - All bases concentrate fire on the selected target.
        - Repeat until all missiles are expended or damage reaches 90% of total potential.

5. **Complexity Analysis**
    - **Initialization:** $O(V^2)$
    - **Path Finding:** $O(V*(V+E))$
    - **Attack:** O(Missiles × Targets)

Scenario 5

In this scenario our main goal is to find a path for weakest missiles first, shoot them through the safe paths and mark night as done, if not find the most effective target and that is the least amount of missiles to get exactly one hit.

Algorithm Workflow

1. **Initialization**
   - **Input:**
     Maximum uncontrolled distance among the available missiles(900km).
   - **Logic**:
     Calculate pairwise distances between all cities
     Connect cities within threshold distance
     Build undirected graph for pathfinding
   - **Output**:
     Connected city network ready for traversal.
2. **Path Finding**
   - **Process:**
     - Identify all base and target vertices from the graph.
     - For each base city, perform Breadth-First Search (BFS) to explore all reachable cities.
     - Record all paths terminating at target cities.
     - For each path, compute:
       - Total distance (sum of edge weights).
       - Spy count (number of cities with spies along the path).
       - Max gap (longest gap between consecutive cities).
   - **Complexity**: O(V*(V+E)) per base (BFS for each base vertex).

3. **Path Organization**
   Categorize paths by missile type, feasibility and safety.
   - **Feasibility**: first check feasibility by checking total distance and comparing max gap with uncontrolled distance.
   - **safety:** A path is **"safe"** if spyCount < missile_stealth. Otherwise, it is **"revealed"**.

4. **Attack Simulation**
   - **Phase 1 (Safe Attacks):**
     - Loop through missiles in priority order
     - If there exists a missile of that type and a safe path for it, shoot it and mark night as done
     -
   - **Phase 2 (Fallback Attacks)**
     - Find the most effective target
     - Find all the fallback bases
     - Execute minimal attack on each target
     - Find the least amount of missiles to get 1 hit

5. **Complexity Analysis**
   - **Initialization:** O(V^2)
   - **Path Finding:** O(V*(V+E))
   - **Attack:** O(Paths × Targets)

Scenario 6

This scenario uses the scenario 5 logic, it first deploys safe attacks each night, shoots 1 missile using safe paths and tries to break the defence of a city using as little missiles as possible to just mark this night as done and safe missiles for later use.

Algorithm Workflow

perform scenario 5 algorithm in a loop till it can no longer attack.

Scenario 7

In this scenario we have different types of missiles including count and its price. Our goal is to reach the desired damage in 7 consecutive nights if so, the enemy will surrender.

Algorithm Workflow

1. **Initialization**
    - **Input:**
      Maximum uncontrolled distance among the available missiles(900km).
    - **Logic**:
      Calculate pairwise distances between all cities
      Connect cities within threshold distance
      Build undirected graph for pathfinding
    - **Output**:
      Connected city network ready for traversal.
2. **Path Finding**
    - **Process:**
        - Identify all base and target vertices from the graph.
        - For each base city, perform Breadth-First Search (BFS) to explore all reachable cities.
        - Record all paths terminating at target cities.
        - For each path, compute:
            - Total distance (sum of edge weights).
            - Spy count (number of cities with spies along the path).
            - Max gap (longest gap between consecutive cities).
    - **Complexity**: O(V*(V+E)) per base (BFS for each base vertex).

3. **Path Organization**
   Categorize paths by missile type, feasibility and safety.
    - **Feasibility**: first check feasibility by checking total distance and comparing max gap with uncontrolled distance.
    - **safety:** A path is **"safe"** if spyCount < missile_stealth. Otherwise, it is **"revealed"**.

4. **Attack simulation**
    - **Phase 1 (Safe Attacks)**
        - Gather all the missiles which have a safe path.
        - Find minimum cost to reach desired damage with those usable missiles
        - If successful launch attack and update inventory
    - **Phase 2 (Fallback Attacks)**
        - For each target gather all the missiles that can reach that target whether it is safe or revealed
        - Sort revealed missiles in damage descending order.
        - Breach the defense with highest damage missiles
        - Add the rest of the revealed missiles to usable missiles.
        - Find minimum cost
        - track what is the lowest cost
        - if successful, launch attack and update inventory, if not mission failed
5. **Minimum Cost**
   This algorithm is similar to a bounded knapsack in which we have a list of items with weight(cost) and value(damage) and knapsack of size W(desired damage).
   Unlike the 0/1 knapsack which we have only one option to take or not to take, but in this variation we can have multiple similar items which can be from 0 up to the amount of missiles of that kind.
    - We use 0/1 knapsack logic but if done naively we get time complexity of **O(missiles * desired damage * count)**
      which is insanely high for large inputs
    - **Use Binary Decomposition**
        - In this way we create group of missiles in which every group has power of 2 items in it
        - The time complexity is reduced to **O(missiles * desired damage * log(count))** which is better than naive approach

6. **Complexity Analysis**
    - **Initialization:** O(V^2)
    - **Path Finding:** O(V*(V+E))
    - **Find Minimum Cost** O(missiles(up to 9) * desired damage * log(count))
    - **Attack:** O(nights * (Find Minimum Cost + target * Find Minimum Cost))