Rescue Drone Task

PLANNING AND APPROXIMATE REASONING

Joan Caballero Castro
Marc Gonzalez Vidal

Master in Artificial Intelligence
Universitat Politècnica de Catalunya (UPC)
Universitat Rovira i Virgili (URV)

Quadrimester 1, 2024/25

Contents

1 Introduction to the Problem

In this assignment, we programmed a rescue drone to navigate a disaster site. The primary objective of the drone is to rescue stranded individuals by transporting them to a designated safe zone. The disaster site is represented by a grid composed of areas, some of which contain obstacles that block the drone's movement. The drone must efficiently plan a sequence of actions to navigate through the grid, avoiding obstacles and reaching the stranded individuals.

In addition to the obstacles, the safe zone has a limited capacity, meaning that it can hold only a certain number of people at any given time. This introduces a planning constraint where the drone must account for the capacity of the safe zone when performing rescue operations. The drone must ensure that all stranded people are rescued while adhering to the capacity constraint of the safe zone.

1.1 Assumptions

In the process of programming this assignment, we made several assumptions to address aspects that were not explicitly defined in the problem:

- When a stranded person is rescued and arrives at the safe zone, they are treated immediately.
- Once treated, individuals remain in the safe zone until the drone issues a command to empty the zone. This simulates the scenario where patients are transported elsewhere after treatment. The drone must be physically present at the safe zone to issue this command, and when the command is given, all treated individuals leave the safe zone at the same time.
- No individual is initially located at the safe zone or in an obstacle.
- All tasks and actions performed by the drone are assumed to have the same cost.
- It is possible for more than one person to be located at the same position on the grid.
- The drone can carry only one person at a time.

2 Analysis of the Problem

2.1 Objects

The following objects represent the entities in the environment that are involved in the rescue operation.

- **Location:** Represents the different points on the grid where the drone can move, pick up people, and drop them off. Locations can also be designated as safe zones or contain obstacles that the drone must avoid.
- **Person:** Represents individuals who are stranded at specific locations in the environment.

2.2 Search Space

To accurately calculate the search space, we need to consider the various combinations of possible states that can arise based on our problem representation. Each state in our problem is defined by the following factors: the total number of locations (L), the number of obstacles (O), the number of stranded persons (P), and the maximum capacity of the safe zone (*max_safe_zone_capacity*).

We will analyze these components step by step to define the final search space formula, which provides the total number of states our system can have.

First, the drone can be located at any of the obstacle-free locations. This means the drone can occupy any of the L - O locations. Thus, the number of possible states for the drone's location is:

$$L - O$$

Next, we need to consider where each stranded person can be located. Each person can either be at one of the obstacle-free locations or be carried by the drone. Therefore, for each person, there are (L - O + *1*) possibilities (with the extra 1 representing the state where the person is being carried by the drone). Since there are (P) stranded persons, the total number of person-location combinations is:

$$(L - O + 1)^P$$

Finally, we must account for the possible capacities of the safe zone. The capacity can range from 0 to *max_safe_zone_capacity*, which gives us:

$$\text{max\_safe\_zone\_capacity} + 1$$

Thus, the total search space is the product of these three components, representing the drone's location, the possible locations of the stranded persons, and the safe zone capacity. The complete formula for the search space is:

$$S = (L - O) \times (L - O + 1)^P \times (\text{max\_safe\_zone\_capacity} + 1)$$

2.3 Operators

The following operators define the actions that the drone can perform to accomplish the rescue mission.

- **move (?d1 - Location ?d2 - Location):** This action moves the drone from its current location (`d1`) to an adjacent location (`d2`). It ensures that the drone only moves to locations that do not contain obstacles.
- **pick-up (?p - Person ?d - Location):** This action allows the drone to pick up a stranded person (`p`) from a specific location (`d`). It requires that the drone and the person are at the same location, and that the drone is not already carrying someone. After execution, the state is updated to indicate that the drone is now carrying the person, and the person is no longer at the original location.
- **drop-off (?p - Person ?d - Location):** This action allows the drone to drop off a carried person (`p`) at the designated safe zone (`d`). It requires the drone to be at the safe zone, carrying the person, and that the safe zone has sufficient capacity. The action marks the person as rescued and updates the capacity of the safe zone accordingly.
- **clean-safe-zone (?d - Location):** This action resets the capacity of the safe zone at location (`d`), clearing it to accommodate more stranded people in the future. The drone must be at the safe zone location for this action to be executed.

2.4 Predicates

The following predicates are used to represent the state of the environment and the relationships between objects.

- **drone-location (?d - Location):** Represents that the drone is currently at the location `d`.
- **person-location (?p - Person ?d - Location):** Indicates that the person `p` is stranded at the location `d`.
- **obstacle (?d - Location):** Indicates that the location `d` contains an obstacle.
- **safe-zone (?d - Location):** Represents that the location `d` is the designated safe zone.
- **adjacent (?d1 - Location ?d2 - Location):** Indicates that the locations `d1` and `d2` are adjacent.

- **rescued (?p - Person):** Represents that the person `p` has been successfully rescued and arrived at the safe zone.
- **carrying (?p - Person):** Indicates that the drone is currently carrying the person `p`.
- **is-carrying-person:** Represents that the drone is currently carrying someone.

2.5 Functions

The following functions are used to represent numerical aspects of the environment, such as capacities and resource management.

- **safe-zone-capacity:** Represents the current number of people in the safe zone. This function helps ensure that the capacity limit of the safe zone is not exceeded during drop-off actions.
- **max-safe-zone-capacity:** Indicates the maximum number of people that the safe zone can accommodate at any given time.

3 PDDL Implementation

```
(define (domain rescue-drone)

(:requirements :strips :typing :fluents :adl)

(:types Location Person - object)

(:predicates
    (drone-location ?d - Location)
    (person-location ?p - Person ?d - Location)
    (obstacle ?d - Location)
    (safe-zone ?d - Location)
    (adjacent ?d1 - Location ?d2 - Location)
    (rescued ?p - Person)
    (carrying ?p - Person)
    (is-carrying-person)
)

(:functions
    (safe-zone-capacity)
    (max-safe-zone-capacity)
)

(:action move ; Moves the drone from location d1 to d2
    :parameters (?d1 - Location ?d2 - Location)
    :precondition (and (drone-location ?d1) (adjacent ?d1 ?d2) (not
        (obstacle ?d2)))
    :effect (and (not (drone-location ?d1)) (drone-location ?d2))
)

(:action pick-up ; Picks up person p from location d
    :parameters (?p - Person ?d - Location)
    :precondition (and (drone-location ?d) (person-location ?p ?d)
        (not (is-carrying-person)))
    :effect (and (not (person-location ?p ?d)) (carrying ?p)
        (is-carrying-person))
)
```

```
(:action drop-off ; Drops-off person p at the safe zone d
    :parameters (?p - Person ?d - Location)
    :precondition (and (drone-location ?d) (safe-zone ?d) (carrying
        ?p) (< (safe-zone-capacity) (max-safe-zone-capacity)))
    :effect (and (rescued ?p) (not (carrying ?p)) (not
        (is-carrying-person)) (increase (safe-zone-capacity) 1))
)

(:action clean-safe-zone ; Empties the safe zone
    :parameters (?d - Location)
    :precondition (and (drone-location ?d) (safe-zone ?d))
    :effect (assign (safe-zone-capacity) 0)
)
)
```

4 Testing Cases

4.1 Test Case 1

Description

This test case represents a 4x4 grid where the drone must rescue 3 stranded people and bring them to the safe zone. The safe zone is at the bottom-left corner of the grid and the drone starts at the upper-left corner. The safe zone has a capacity of 3 people, so there is no need to empty the safe zone.

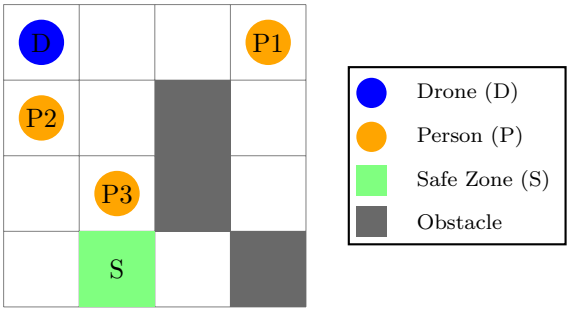

Figure 1: Initial State Representation of Test Case 1

The objective of this test case is to verify that fundamental constraints are respected, such as preventing the drone from moving through obstacles and ensuring it only carries one person at a time. Additionally, this test aims to confirm that the sequence of actions is executed correctly, with the drone navigating efficiently through the grid while picking up and dropping off individuals at the safe zone.

Problem

Here is the definition of the PDDL problem for this test case, representing the initial state of the environment and the goal.

```
(define (problem problem1)
(:domain rescue-drone)

(:objects
    d1 d2 d3 d4 d5 d6 d7 d8 d9 d10 d11 d12 d13 d14 d15 d16 - Location
    p1 p2 p3 - Person
)
```

```
(:init
    (= (safe-zone-capacity) 0)
    (= (max-safe-zone-capacity) 3)

    ; Row 1: d1 to d4
    (adjacent d1 d2) (adjacent d1 d5)
    (adjacent d2 d1) (adjacent d2 d3) (adjacent d2 d6)
    (adjacent d3 d2) (adjacent d3 d4) (adjacent d3 d7)
    (adjacent d4 d3) (adjacent d4 d8)

    ; Row 2: d5 to d8
    (adjacent d5 d1) (adjacent d5 d6) (adjacent d5 d9)
    (adjacent d6 d2) (adjacent d6 d5) (adjacent d6 d7) (adjacent d6
        d10)
    (adjacent d7 d3) (adjacent d7 d6) (adjacent d7 d8) (adjacent d7
        d11)
    (adjacent d8 d4) (adjacent d8 d7) (adjacent d8 d12)

    ; Row 3: d9 to d12
    (adjacent d9 d5) (adjacent d9 d10) (adjacent d9 d13)
    (adjacent d10 d6) (adjacent d10 d9) (adjacent d10 d11) (adjacent
        d10 d14)
    (adjacent d11 d7) (adjacent d11 d10) (adjacent d11 d12)
        (adjacent d11 d15)
    (adjacent d12 d8) (adjacent d12 d11) (adjacent d12 d16)

    ; Row 4: d13 to d16
    (adjacent d13 d9) (adjacent d13 d14)
    (adjacent d14 d10) (adjacent d14 d13) (adjacent d14 d15)
    (adjacent d15 d11) (adjacent d15 d14) (adjacent d15 d16)
    (adjacent d16 d12) (adjacent d16 d15)

    (drone-location d1)

    (person-location p1 d4)
    (person-location p2 d5)
    (person-location p3 d10)

    (obstacle d7) (obstacle d11) (obstacle d16)

    (safe-zone d14)
)

(:goal (forall (?p - Person) (rescued ?p)))
)
```

Results

The following plan was generated by the planner for this test case. Each step represents an action taken by the drone to achieve the goal.

```
step   0: MOVE D1 D5
       1: PICK-UP P2 D5
       2: MOVE D5 D6
       3: MOVE D6 D10
       4: MOVE D10 D14
       5: DROP-OFF P2 D14
       6: MOVE D14 D10
       7: PICK-UP P3 D10
       8: MOVE D10 D14
       9: DROP-OFF P3 D14
      10: MOVE D14 D10
      11: MOVE D10 D6
      12: MOVE D6 D2
      13: MOVE D2 D3
      14: MOVE D3 D4
      15: PICK-UP P1 D4
      16: MOVE D4 D3
      17: MOVE D3 D2
      18: MOVE D2 D6
      19: MOVE D6 D10
      20: MOVE D10 D14
      21: DROP-OFF P1 D14
```

Analysis

The drone begins by moving to the closest person (P2), picks them up, and takes the most efficient route to the safe zone. It then proceeds to the next closest person (P3), picks them up, and transports them to the safe zone. Finally, the drone returns via the shortest path to P2, picks them up again, and drops them off at the safe zone.

We observe that the drone consistently prioritizes rescuing the nearest individual first and immediately transports them to the safe zone. Throughout this test case, the drone followed the shortest possible paths, resulting in the most efficient plan compared to other scenarios.

4.2 Test Case 2

Description

This test case represents a 5x5 grid where the drone must rescue 6 stranded people and bring them to the safe zone. The safe zone is at the bottom-right corner of the grid and the drone starts at the upper-left corner. The safe zone has a capacity of 3 people, so it is necessary to empty the safe zone to accommodate all stranded people.

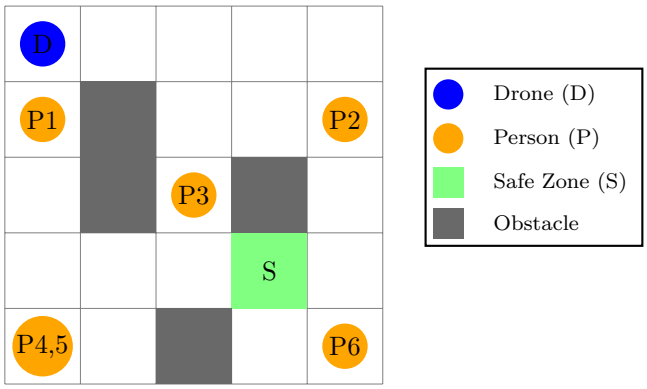

Figure 2: Initial State Representation of Test Case 2

The objective of this test case is to assess the drone's performance in a larger and more complex grid environment. Specifically, we aim to determine whether the drone can navigate efficiently through the increased complexity and adhere to the constraint of carrying only one person at a time, even when multiple individuals are present in the same grid cell. Furthermore, this test evaluates the drone's ability to manage the safe zone's maximum capacity by requiring it to empty the safe zone when necessary.

Problem

Here is the definition of the PDDL problem for this test case, representing the initial state of the environment and the goal.

```
(define (problem problem2)
(:domain rescue-drone)

(:objects
    d1 d2 d3 d4 d5 d6 d7 d8 d9 d10 d11 d12 d13 d14 d15 d16 d17 d18
        d19 d20 d21 d22 d23 d24 d25 - Location
    p1 p2 p3 p4 p5 p6 - Person
)

(:init
    (= (safe-zone-capacity) 0)
    (= (max-safe-zone-capacity) 3)

    ; Row 1: d1 to d5
    (adjacent d1 d2) (adjacent d1 d6)
    (adjacent d2 d1) (adjacent d2 d3) (adjacent d2 d7)
    (adjacent d3 d2) (adjacent d3 d4) (adjacent d3 d8)
```

```
    (adjacent d4 d3) (adjacent d4 d5) (adjacent d4 d9)
    (adjacent d5 d4) (adjacent d5 d10)

    ; Row 2: d6 to d10
    (adjacent d6 d1) (adjacent d6 d7) (adjacent d6 d11)
    (adjacent d7 d2) (adjacent d7 d6) (adjacent d7 d8) (adjacent d7
        d12)
    (adjacent d8 d3) (adjacent d8 d7) (adjacent d8 d9) (adjacent d8
        d13)
    (adjacent d9 d4) (adjacent d9 d8) (adjacent d9 d10) (adjacent d9
        d14)
    (adjacent d10 d5) (adjacent d10 d9) (adjacent d10 d15)

    ; Row 3: d11 to d15
    (adjacent d11 d6) (adjacent d11 d12) (adjacent d11 d16)
    (adjacent d12 d7) (adjacent d12 d11) (adjacent d12 d13)
        (adjacent d12 d17)
    (adjacent d13 d8) (adjacent d13 d12) (adjacent d13 d14)
        (adjacent d13 d18)
    (adjacent d14 d9) (adjacent d14 d13) (adjacent d14 d15)
        (adjacent d14 d19)
    (adjacent d15 d10) (adjacent d15 d14) (adjacent d15 d20)

    ; Row 4: d16 to d20
    (adjacent d16 d11) (adjacent d16 d17) (adjacent d16 d21)
    (adjacent d17 d12) (adjacent d17 d16) (adjacent d17 d18)
        (adjacent d17 d22)
    (adjacent d18 d13) (adjacent d18 d17) (adjacent d18 d19)
        (adjacent d18 d23)
    (adjacent d19 d14) (adjacent d19 d18) (adjacent d19 d20)
        (adjacent d19 d24)
    (adjacent d20 d15) (adjacent d20 d19) (adjacent d20 d25)

    ; Row 5: d21 to d25
    (adjacent d21 d16) (adjacent d21 d22)
    (adjacent d22 d17) (adjacent d22 d21) (adjacent d22 d23)
    (adjacent d23 d18) (adjacent d23 d22) (adjacent d23 d24)
    (adjacent d24 d19) (adjacent d24 d23) (adjacent d24 d25)
    (adjacent d25 d20) (adjacent d25 d24)

    (drone-location d1)

    (person-location p1 d6) (person-location p2 d10)
    (person-location p3 d13) (person-location p4 d21)
    (person-location p5 d21) (person-location p6 d25)

    (obstacle d7) (obstacle d12)
    (obstacle d14) (obstacle d23)

    (safe-zone d19)
)
```



```
(:goal (forall (?p - Person) (rescued ?p)))
)
```

Results

The following plan was generated by the planner for this test case. Each step represents an action taken by the drone to achieve the goal.

```
step   0: MOVE D1 D6
       1: PICK-UP P1 D6
       2: MOVE D6 D11
       3: MOVE D11 D16
       4: MOVE D16 D17
       5: MOVE D17 D18
       6: MOVE D18 D19
       7: DROP-OFF P1 D19
       8: MOVE D19 D20
       9: MOVE D20 D25
      10: PICK-UP P6 D25
      11: MOVE D25 D20
      12: MOVE D20 D19
      13: DROP-OFF P6 D19
      14: MOVE D19 D18
      15: MOVE D18 D13
      16: PICK-UP P3 D13
      17: MOVE D13 D18
      18: MOVE D18 D19
      19: DROP-OFF P3 D19
      20: CLEAN-SAFE-ZONE D19
      21: MOVE D19 D20
      22: MOVE D20 D15
      23: MOVE D15 D10
      24: PICK-UP P2 D10
      25: MOVE D10 D15
      26: MOVE D15 D20
      27: MOVE D20 D19
      28: DROP-OFF P2 D19
      29: MOVE D19 D18
      30: MOVE D18 D17
      31: MOVE D17 D22
      32: MOVE D22 D21
      33: PICK-UP P5 D21
      34: MOVE D21 D22
      35: MOVE D22 D17
      36: MOVE D17 D18
      37: MOVE D18 D19
      38: DROP-OFF P5 D19
      39: MOVE D19 D18
      40: MOVE D18 D17
      41: MOVE D17 D22
      42: MOVE D22 D21
      43: PICK-UP P4 D21
      44: MOVE D21 D22
      45: MOVE D22 D17
      46: MOVE D17 D18
      47: MOVE D18 D19
      48: DROP-OFF P4 D19
```

Analysis

In this scenario, the drone starts by picking up the closest person (P1) and transporting them to the safe zone via the shortest route. It then rescues the next closest person (P6), followed by P3, each time using the most efficient paths. Once the drone has dropped off these three individuals, the safe zone reaches its maximum capacity. Consequently, the drone empties the safe zone immediately after dropping off the last person. Subsequently, the drone continues to rescue the remaining individuals (P2, P5, and P4), always following the closest paths for pickup and drop-off.

We observe that every time the drone departs from the safe zone to pick up someone, it consistently follows the same path both to and from the safe zone, ensuring efficiency. Additionally, the drone always selects the shortest possible routes, resulting in the most efficient plan among all tested scenarios.

4.3 Test Case 3

Description

This test case represents a 5x7 grid where the drone must rescue 10 stranded people and bring them to the safe zone. The safe zone is at the bottom-right corner of the grid and the drone starts at the upper-left corner. The safe zone has a capacity of 3 people, so it is necessary to empty the safe zone to accommodate all stranded people.

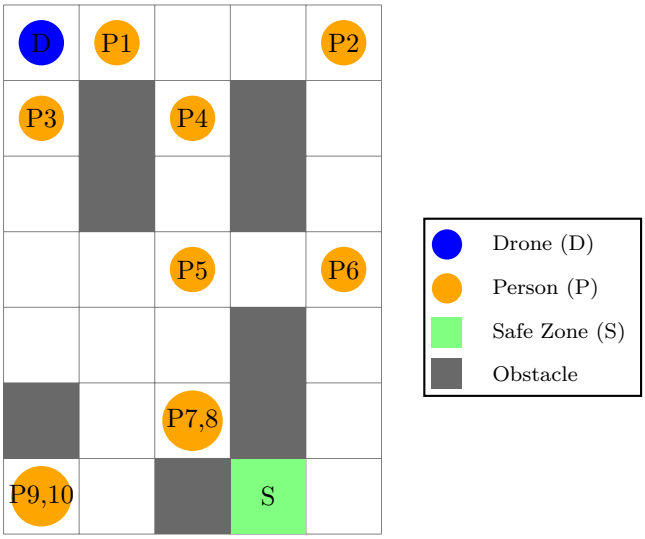

Figure 3: Initial State Representation of Test Case 3

The objective of this test case is to evaluate the drone's performance when rescuing a large number of individuals in an extensive environment. This includes analyzing the drone's movement patterns for picking up and dropping off people across a large grid with numerous obstacles, which significantly reduce the available paths. The test aims to determine whether the drone can effectively navigate the complex grid while maintaining efficient rescue operations.

Problem

Here is the definition of the PDDL problem for this test case, representing the initial state of the environment and the goal.

```
(define (problem problem3)
(:domain rescue-drone)

(:objects
    d1 d2 d3 d4 d5 d6 d7 d8 d9 d10 d11 d12 d13 d14 d15 d16 d17 d18
        d19 d20 d21 d22 d23 d24 d25 d26 d27 d28 d29 d30 d31 d32 d33
        d34 d35 - Location
    p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 - Person
)

(:init
    (= (safe-zone-capacity) 0)
    (= (max-safe-zone-capacity) 3)

    ; Row 1: d1 to d5
    (adjacent d1 d2) (adjacent d1 d6)
    (adjacent d2 d1) (adjacent d2 d3) (adjacent d2 d7)
    (adjacent d3 d2) (adjacent d3 d4) (adjacent d3 d8)
    (adjacent d4 d3) (adjacent d4 d5) (adjacent d4 d9)
    (adjacent d5 d4) (adjacent d5 d10)

    ; Row 2: d6 to d10
    (adjacent d6 d1) (adjacent d6 d7) (adjacent d6 d11)
    (adjacent d7 d2) (adjacent d7 d6) (adjacent d7 d8) (adjacent d7
        d12)
    (adjacent d8 d3) (adjacent d8 d7) (adjacent d8 d9) (adjacent d8
        d13)
    (adjacent d9 d4) (adjacent d9 d8) (adjacent d9 d10) (adjacent d9
        d14)
    (adjacent d10 d5) (adjacent d10 d9) (adjacent d10 d15)

    ; Row 3: d11 to d15
    (adjacent d11 d6) (adjacent d11 d12) (adjacent d11 d16)
    (adjacent d12 d7) (adjacent d12 d11) (adjacent d12 d13)
        (adjacent d12 d17)
    (adjacent d13 d8) (adjacent d13 d12) (adjacent d13 d14)
        (adjacent d13 d18)
    (adjacent d14 d9) (adjacent d14 d13) (adjacent d14 d15)
        (adjacent d14 d19)
    (adjacent d15 d10) (adjacent d15 d14) (adjacent d15 d20)

    ; Row 4: d16 to d20
    (adjacent d16 d11) (adjacent d16 d17) (adjacent d16 d21)
    (adjacent d17 d12) (adjacent d17 d16) (adjacent d17 d18)
        (adjacent d17 d22)
    (adjacent d18 d13) (adjacent d18 d17) (adjacent d18 d19)
        (adjacent d18 d23)
```

```
    (adjacent d19 d14) (adjacent d19 d18) (adjacent d19 d20)
        (adjacent d19 d24)
    (adjacent d20 d15) (adjacent d20 d19) (adjacent d20 d25)

    ; Row 5: d21 to d25
    (adjacent d21 d16) (adjacent d21 d22) (adjacent d21 d26)
    (adjacent d22 d17) (adjacent d22 d21) (adjacent d22 d23)
        (adjacent d22 d27)
    (adjacent d23 d18) (adjacent d23 d22) (adjacent d23 d24)
        (adjacent d23 d28)
    (adjacent d24 d19) (adjacent d24 d23) (adjacent d24 d25)
        (adjacent d24 d29)
    (adjacent d25 d20) (adjacent d25 d24) (adjacent d25 d30)

    ; Row 6: d26 to d30
    (adjacent d26 d21) (adjacent d26 d27) (adjacent d26 d31)
    (adjacent d27 d22) (adjacent d27 d26) (adjacent d27 d28)
        (adjacent d27 d32)
    (adjacent d28 d23) (adjacent d28 d27) (adjacent d28 d29)
        (adjacent d28 d33)
    (adjacent d29 d24) (adjacent d29 d28) (adjacent d29 d30)
        (adjacent d29 d34)
    (adjacent d30 d25) (adjacent d30 d29) (adjacent d30 d35)

    ; Row 7: d31 to d35
    (adjacent d31 d26) (adjacent d31 d32)
    (adjacent d32 d27) (adjacent d32 d31) (adjacent d32 d33)
    (adjacent d33 d28) (adjacent d33 d32) (adjacent d33 d34)
    (adjacent d34 d29) (adjacent d34 d33) (adjacent d34 d35)
    (adjacent d35 d30) (adjacent d35 d34)

    (drone-location d1)

    (person-location p1 d2) (person-location p2 d5)
    (person-location p3 d6) (person-location p4 d8)
    (person-location p5 d18) (person-location p6 d20)
    (person-location p7 d28) (person-location p8 d28)
    (person-location p9 d31) (person-location p10 d31)

    (obstacle d7) (obstacle d9) (obstacle d12) (obstacle d14)
    (obstacle d24) (obstacle d26) (obstacle d29) (obstacle d33)

    (safe-zone d34)
)

(:goal (forall (?p - Person) (rescued ?p)))
)
```

Results

The following plan was generated by the planner for this test case. Each step represents an action taken by the drone to achieve the goal.

```
step   0: MOVE D1 D2
       1: PICK-UP P1 D2
       2: MOVE D2 D3
       3: MOVE D3 D4
       4: MOVE D4 D5
       5: MOVE D5 D10
       6: MOVE D10 D15
       7: MOVE D15 D20
       8: MOVE D20 D25
       9: MOVE D25 D30
      10: MOVE D30 D35
      11: MOVE D35 D34
      12: DROP-OFF P1 D34
      13: MOVE D34 D35
      14: MOVE D35 D30
      15: MOVE D30 D25
      16: MOVE D25 D20
      17: MOVE D20 D15
      18: MOVE D15 D10
      19: MOVE D10 D5
      20: PICK-UP P2 D5
      21: MOVE D5 D10
      22: MOVE D10 D15
      23: MOVE D15 D20
      24: MOVE D20 D25
      25: MOVE D25 D30
      26: MOVE D30 D35
      27: MOVE D35 D34
      28: DROP-OFF P2 D34
      29: MOVE D34 D35
      30: MOVE D35 D30
      31: MOVE D30 D25
      32: MOVE D25 D20
      33: MOVE D20 D19
      34: MOVE D19 D18
      35: MOVE D18 D13
      36: MOVE D13 D8
      37: PICK-UP P4 D8
      38: MOVE D8 D13
      39: MOVE D13 D18
      40: MOVE D18 D19
      41: MOVE D19 D20
      42: MOVE D20 D25
      43: MOVE D25 D30
      44: MOVE D30 D35
      45: MOVE D35 D34
      46: DROP-OFF P4 D34
      47: CLEAN-SAFE-ZONE D34
      48: MOVE D34 D35
      49: MOVE D35 D30
      50: MOVE D30 D25
      51: MOVE D25 D20
      52: MOVE D20 D19
      53: MOVE D19 D18
      54: MOVE D18 D17
      55: MOVE D17 D16
      56: MOVE D16 D11
      57: MOVE D11 D6
      58: PICK-UP P3 D6
      59: MOVE D6 D11
      60: MOVE D11 D16
      61: MOVE D16 D17
      62: MOVE D17 D18
      63: MOVE D18 D19
      64: MOVE D19 D20
      65: MOVE D20 D25
      66: MOVE D25 D30
      67: MOVE D30 D35
      68: MOVE D35 D34
      69: DROP-OFF P3 D34
      70: MOVE D34 D35
      71: MOVE D35 D30
      72: MOVE D30 D25
      73: MOVE D25 D20
      74: MOVE D20 D19
      75: MOVE D19 D18
      76: MOVE D18 D23
      77: MOVE D23 D28
      78: MOVE D28 D27
      79: MOVE D27 D32
      80: MOVE D32 D31
      81: PICK-UP P9 D31
```

```
 82: MOVE D31 D32
 83: MOVE D32 D27
 84: MOVE D27 D28
 85: MOVE D28 D23
 86: MOVE D23 D18
 87: MOVE D18 D19
 88: MOVE D19 D20
 89: MOVE D20 D25
 90: MOVE D25 D30
 91: MOVE D30 D35
 92: MOVE D35 D34
 93: DROP-OFF P9 D34
 94: MOVE D34 D35
 95: MOVE D35 D30
 96: MOVE D30 D25
 97: MOVE D25 D20
 98: MOVE D20 D19
 99: MOVE D19 D18
100: MOVE D18 D23
101: MOVE D23 D28
102: MOVE D28 D27
103: MOVE D27 D32
104: MOVE D32 D31
105: PICK-UP P10 D31
106: MOVE D31 D32
107: MOVE D32 D27
108: MOVE D27 D28
109: MOVE D28 D23
110: MOVE D23 D18
111: MOVE D18 D19
112: MOVE D19 D20
113: MOVE D20 D25
114: MOVE D25 D30
115: MOVE D30 D35
116: MOVE D35 D34
117: DROP-OFF P10 D34
118: CLEAN-SAFE-ZONE D34
119: MOVE D34 D35
120: MOVE D35 D30
121: MOVE D30 D25
122: MOVE D25 D20
123: MOVE D20 D19
124: MOVE D19 D18
125: MOVE D18 D23
126: MOVE D23 D28
127: PICK-UP P7 D28
128: MOVE D28 D23
129: MOVE D23 D18
130: MOVE D18 D19
```

```
131: MOVE D19 D20
132: MOVE D20 D25
133: MOVE D25 D30
134: MOVE D30 D35
135: MOVE D35 D34
136: DROP-OFF P7 D34
137: MOVE D34 D35
138: MOVE D35 D30
139: MOVE D30 D25
140: MOVE D25 D20
141: MOVE D20 D19
142: MOVE D19 D18
143: MOVE D18 D23
144: MOVE D23 D28
145: PICK-UP P8 D28
146: MOVE D28 D23
147: MOVE D23 D18
148: MOVE D18 D19
149: MOVE D19 D20
150: MOVE D20 D25
151: MOVE D25 D30
152: MOVE D30 D35
153: MOVE D35 D34
154: DROP-OFF P8 D34
155: MOVE D34 D35
156: MOVE D35 D30
157: MOVE D30 D25
158: MOVE D25 D20
159: MOVE D20 D19
160: MOVE D19 D18
161: PICK-UP P5 D18
162: MOVE D18 D19
163: MOVE D19 D20
164: MOVE D20 D25
165: MOVE D25 D30
166: MOVE D30 D35
167: MOVE D35 D34
168: DROP-OFF P5 D34
169: CLEAN-SAFE-ZONE D34
170: MOVE D34 D35
171: MOVE D35 D30
172: MOVE D30 D25
173: MOVE D25 D20
174: PICK-UP P6 D20
175: MOVE D20 D25
176: MOVE D25 D30
177: MOVE D30 D35
178: MOVE D35 D34
179: DROP-OFF P6 D34
```

Analysis

In this scenario, the drone does not follow a strict order when selecting which person to pick up first, unlike the previous test cases where it always chose the closest individual. Despite the lack of a specific order, the drone consistently picks up and drops off individuals using the shortest possible paths. Each time the safe zone reaches its maximum capacity, the drone promptly clears the safe zone after dropping off the third person, similar to the previous test case. While we did not analyze all possible path combinations, comparing this test case with plans from other students indicates that the drone achieved one of the shortest plan lengths.

Although the drone does not prioritize the closest individual first, it maintains efficiency by always selecting the shortest paths for pickup and drop-off actions. The consistent clearing of the safe zone upon reaching maximum capacity ensures that all individuals are eventually rescued. This test case demonstrates the drone's ability to handle complex environments and multiple constraints while maintaining an efficient rescue plan.

4.4 Test Case 4

Description

This test case represents a 3x3 grid where the drone must rescue 5 stranded people and bring them to the safe zone. The safe zone is at the bottom-left corner of the grid and the drone starts at the upper-left corner. The safe zone has a capacity of 2 people, so it is necessary to empty the safe zone to accommodate all stranded people.

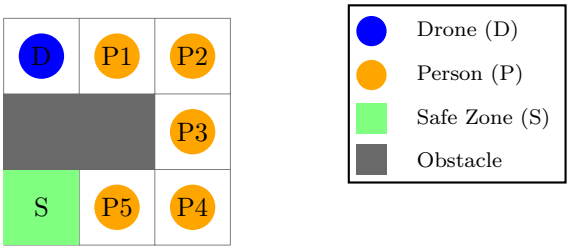

Figure 4: Initial State Representation of Test Case 4

The objective of this test case is to examine the drone's performance in a small grid with a high number of stranded individuals. This scenario tests the drone's ability to manage multiple rescue operations within a confined space, efficiently navigating and rescuing all individuals without violating any constraints.

Problem

Here is the definition of the PDDL problem for this test case, representing the initial state of the environment and the goal.

```
(define (problem problem4)
(:domain rescue-drone)

(:objects
    d1 d2 d3 d4 d5 d6 d7 d8 d9 - Location
    p1 p2 p3 p4 p5 - Person
)

(:init
    (= (safe-zone-capacity) 0)
    (= (max-safe-zone-capacity) 2)

    (adjacent d1 d2) (adjacent d1 d4)
    (adjacent d2 d1) (adjacent d2 d3) (adjacent d2 d5)
    (adjacent d3 d2) (adjacent d3 d6)
    (adjacent d4 d1) (adjacent d4 d5) (adjacent d4 d7)
    (adjacent d5 d2) (adjacent d5 d4) (adjacent d5 d6) (adjacent d5
        d8)
    (adjacent d6 d3) (adjacent d6 d5) (adjacent d6 d9)
    (adjacent d7 d4) (adjacent d7 d8)
    (adjacent d8 d5) (adjacent d8 d7) (adjacent d8 d9)
    (adjacent d9 d6) (adjacent d9 d8)

    (drone-location d1)

    (person-location p1 d2) (person-location p2 d3)
    (person-location p3 d6) (person-location p4 d9)
    (person-location p5 d8)

    (obstacle d4) (obstacle d5)

    (safe-zone d7)
)

(:goal (forall (?p - Person) (rescued ?p)))
)
```

Results

The following plan was generated by the planner for this test case. Each step represents an action taken by the drone to achieve the goal.

```
step   0: MOVE D1 D2
       1: PICK-UP P1 D2
       2: MOVE D2 D3
       3: MOVE D3 D6
       4: MOVE D6 D9
       5: MOVE D9 D8
       6: MOVE D8 D7
       7: DROP-OFF P1 D7
       8: MOVE D7 D8
       9: PICK-UP P5 D8
      10: MOVE D8 D7
      11: DROP-OFF P5 D7
      12: CLEAN-SAFE-ZONE D7
      13: MOVE D7 D8
      14: MOVE D8 D9
      15: PICK-UP P4 D9
      16: MOVE D9 D8
      17: MOVE D8 D7
      18: DROP-OFF P4 D7
```

```
      19: MOVE D7 D8
      20: MOVE D8 D9
      21: MOVE D9 D6
      22: PICK-UP P3 D6
      23: MOVE D6 D9
      24: MOVE D9 D8
      25: MOVE D8 D7
      26: DROP-OFF P3 D7
      27: CLEAN-SAFE-ZONE D7
      28: MOVE D7 D8
      29: MOVE D8 D9
      30: MOVE D9 D6
      31: MOVE D6 D3
      32: PICK-UP P2 D3
      33: MOVE D3 D6
      34: MOVE D6 D9
      35: MOVE D9 D8
      36: MOVE D8 D7
      37: DROP-OFF P2 D7
```

Analysis

In this scenario, the drone begins by picking up the closest person (P1) and transporting them to the safe zone. It then continues to rescue the next closest individuals, always following the shortest possible paths for each pickup and drop-off. Whenever the safe zone reaches its maximum capacity of two persons, the drone immediately empties the safe zone after dropping off the last individual. This process is repeated until all stranded individuals are rescued.

We observe that the drone consistently prioritizes the nearest available person and follows the shortest routes for all movements. Each time the safe zone reaches its capacity, the drone promptly clears it, ensuring that the safe zone remains available for subsequent rescues. As a result, the drone successfully follows the shortest plan for this test case, demonstrating its ability to efficiently handle multiple rescues in a limited space.

4.5 Test Case 5

Description

This test case represents a 5x5 grid where the drone must rescue 1 stranded person and bring them to the safe zone. The safe zone is at the center-right of the grid and the drone starts at the center-left. Although the safe zone has a capacity for one person, the stranded individual is located in an inaccessible area, blocked by obstacles. As a result, the drone cannot reach the person, and the problem is unsolvable.

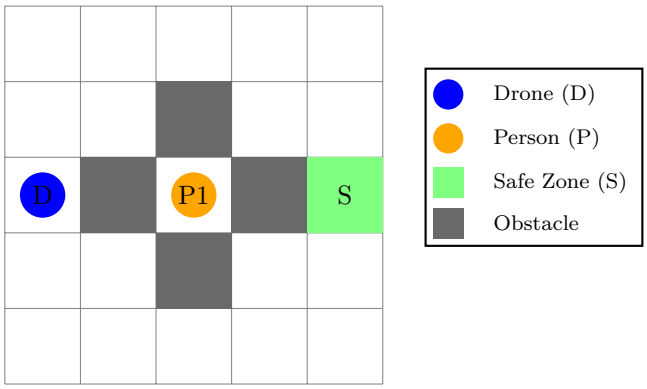

Figure 5: Initial State Representation of Test Case 5

The objective of this test case is to determine whether the planner we are using (the Metric-FF planner) can accurately detect when a test case is unsolvable. Specifically, we aim to verify if the planner can quickly identify that the problem is not satisfiable due to inaccessible locations or other constraints, ensuring that it does not attempt to generate a plan where none exists.

Problem

Here is the definition of the PDDL problem for this test case, representing the initial state of the environment and the goal.

```
(define (problem problem5)
(:domain rescue-drone)

(:objects
    d1 d2 d3 d4 d5 d6 d7 d8 d9 d10 d11 d12 d13 d14 d15 d16 d17 d18
        d19 d20 d21 d22 d23 d24 d25 - Location
    p1 - Person
)

(:init
    (= (safe-zone-capacity) 0)
    (= (max-safe-zone-capacity) 1)

    ; Row 1: d1 to d5
    (adjacent d1 d2) (adjacent d1 d6)
    (adjacent d2 d1) (adjacent d2 d3) (adjacent d2 d7)
    (adjacent d3 d2) (adjacent d3 d4) (adjacent d3 d8)
    (adjacent d4 d3) (adjacent d4 d5) (adjacent d4 d9)
    (adjacent d5 d4) (adjacent d5 d10)
```

```
    ; Row 2: d6 to d10
    (adjacent d6 d1) (adjacent d6 d7) (adjacent d6 d11)
    (adjacent d7 d2) (adjacent d7 d6) (adjacent d7 d8) (adjacent d7
        d12)
    (adjacent d8 d3) (adjacent d8 d7) (adjacent d8 d9) (adjacent d8
        d13)
    (adjacent d9 d4) (adjacent d9 d8) (adjacent d9 d10) (adjacent d9
        d14)
    (adjacent d10 d5) (adjacent d10 d9) (adjacent d10 d15)

    ; Row 3: d11 to d15
    (adjacent d11 d6) (adjacent d11 d12) (adjacent d11 d16)
    (adjacent d12 d7) (adjacent d12 d11) (adjacent d12 d13)
        (adjacent d12 d17)
    (adjacent d13 d8) (adjacent d13 d12) (adjacent d13 d14)
        (adjacent d13 d18)
    (adjacent d14 d9) (adjacent d14 d13) (adjacent d14 d15)
        (adjacent d14 d19)
    (adjacent d15 d10) (adjacent d15 d14) (adjacent d15 d20)

    ; Row 4: d16 to d20
    (adjacent d16 d11) (adjacent d16 d17) (adjacent d16 d21)
    (adjacent d17 d12) (adjacent d17 d16) (adjacent d17 d18)
        (adjacent d17 d22)
    (adjacent d18 d13) (adjacent d18 d17) (adjacent d18 d19)
        (adjacent d18 d23)
    (adjacent d19 d14) (adjacent d19 d18) (adjacent d19 d20)
        (adjacent d19 d24)
    (adjacent d20 d15) (adjacent d20 d19) (adjacent d20 d25)

    ; Row 5: d21 to d25
    (adjacent d21 d16) (adjacent d21 d22)
    (adjacent d22 d17) (adjacent d22 d21) (adjacent d22 d23)
    (adjacent d23 d18) (adjacent d23 d22) (adjacent d23 d24)
    (adjacent d24 d19) (adjacent d24 d23) (adjacent d24 d25)
    (adjacent d25 d20) (adjacent d25 d24)

    (drone-location d11)

    (person-location p1 d13)

    (obstacle d8) (obstacle d12)
    (obstacle d14) (obstacle d18)

    (safe-zone d15)
)

(:goal (forall (?p - Person) (rescued ?p)))
)
```

Results

The planner was unable to generate a plan for this test case.

```
ff: goal can be simplified to FALSE. No plan will solve it
```

Analysis

In this scenario, the drone is tasked with rescuing one stranded person located in an area that is completely inaccessible due to obstacles. The safe zone has a capacity of one person, and despite the drone's efforts, it cannot reach the stranded individual. Consequently, there is no viable solution for this test case.

As expected, the planner did not generate a plan for this test case, correctly identifying it as unsolvable. This demonstrates the planner's ability to detect infeasible scenarios and prevent the generation of invalid plans, thereby ensuring the reliability of the planning process.

5 Analysis of the Results

This section provides a detailed analysis of the results, focusing on key metrics affected by various aspects of the problem. Since our solution involves numeric fluents and types, we utilized the Metric-FF planner, which supports these features. The experiments were conducted locally on an Ubuntu virtual machine.

5.1 Complexity Analysis

This subsection presents the complexity of the solutions found by the planner for each test case. The table below summarizes key factors such as the number of stranded persons, grid size (number of locations), plan length, and the number of nodes expanded. These factors provide deeper insights into the complexity of each test case and how the planner's performance scales with these variables.

While execution time is an important metric in evaluating the performance of planners, it was not included in our analysis due to the difficulty of obtaining precise measurements with our current setup. Both the Metric-FF planner and Ubuntu's internal time system introduce variability caused by factors such as process management, cache interference, and background tasks, which affect the accuracy of time as a metric.

Additionally, small differences in test case configurations can lead to execution time variations that are too minor to be measured accurately. Therefore, we focused on plan length and nodes expanded, which provide more reliable indicators of the planner's performance. Future work could involve developing a more accurate method for measuring execution time to include it as a key metric in such studies.

Test Case	**Locations**	**Persons**	**Plan Length**	**Nodes Expanded**
1	16	3	22	34
2	25	6	49	96
3	35	10	180	390
4	9	5	38	51

Table 1: Comparison of Test Case Characteristics and Complexity Metrics

As shown in Table 1, the complexity of each test case varies based on several factors, including the number of stranded persons, the grid size (number of locations), and the resulting plan length and nodes expanded.

In **Test Case 1**, the small grid (16 locations) and low number of stranded persons (3) result in a short plan length of 22 steps and relatively few nodes expanded (34). This reflects the simplicity of the problem in a limited environment with minimal constraints.

Test Case 2 introduces more complexity with a larger grid (25 locations) and twice as many stranded persons (6). This leads to a longer plan (49) and almost triple the number of nodes expanded (96), indicating that even moderate increases in grid size and the number of persons significantly raise the complexity.

Test Case 3 has the largest grid (35 locations) and the most stranded persons (10), leading to a substantial increase in both plan length (180) and nodes expanded (390). The high complexity in this case stems from the need to navigate a larger environment while rescuing more people, all while avoiding obstacles and managing the safe zone.

In contrast, **Test Case 4** features a smaller grid (9 locations) but a comparable number of stranded persons (5). Despite this, the plan length (38) and nodes expanded (51) are lower than in Test Case 2, indicating that grid size plays a significant role in determining complexity, even when the number of persons is similar.

This analysis highlights how the interplay of grid size and number of stranded persons contributes to the overall complexity of each test case. Larger grids with more stranded persons naturally lead to longer plans and more nodes being expanded, reflecting the planner's increasing computational demands as the problem grows.

5.2 Impact of Stranded Persons on Solution Complexity

In this subsection, we analyze how increasing the ratio of stranded persons to available locations affects the complexity of the solutions. By adjusting the number of stranded persons proportionally to the available grid locations in each test case, we normalize the density of stranded persons across different environments. This approach ensures a fair comparison of the plan length and the number of nodes expanded as the ratio increases, regardless of grid size or obstacle placement.

We present the results in two graphs and their accompanying tables, which illustrate how the plan length and the number of nodes expanded change with the increasing ratio of stranded persons to available locations.

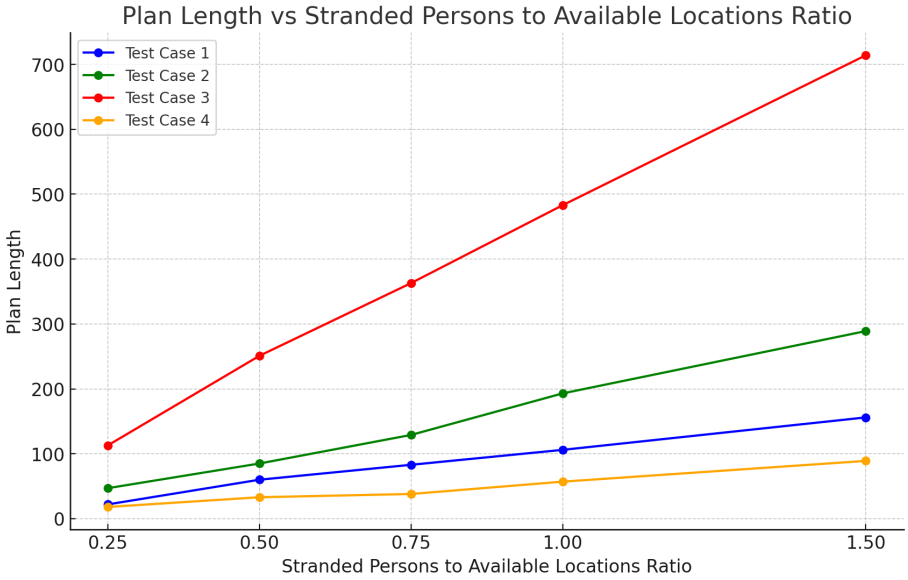

Figure 6: Plan Length Ratio vs Stranded Persons to Available Locations Ratio for Different Test Cases

Ratio	**Plan Length (Number of Stranded Persons)**			
	Test Case 1	**Test Case 2**	**Test Case 3**	**Test Case 4**
0.25	22 (3)	47 (5)	113 (7)	18 (2)
0.5	60 (7)	85 (11)	251 (14)	33 (4)
0.75	83 (10)	129 (16)	363 (20)	38 (5)
1.0	106 (13)	193 (21)	483 (27)	57 (7)
1.5	156 (20)	289 (32)	714 (41)	89 (10)

Table 2: Plan Length at Various Ratios of Stranded Persons to Available Locations

As shown in Figure 6 and Table 2, the plan length increases as the ratio grows. Test Case 3, which has the largest grid and the highest number of stranded persons, demonstrates the most significant increase in plan length. In contrast, Test Cases 1 and 4, with smaller grids and fewer stranded persons, show a more gradual increase. This indicates that the complexity of the solution escalates significantly in larger, more densely populated grids.

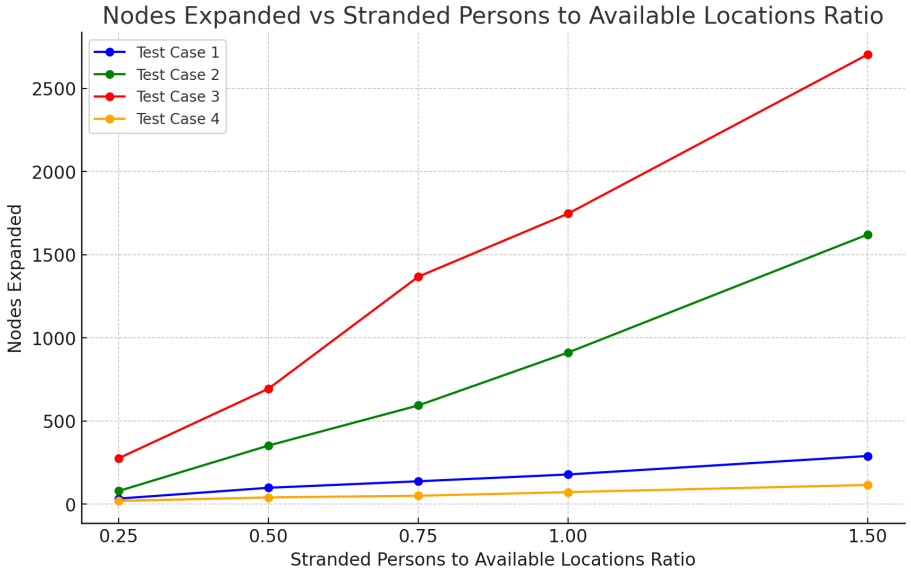

Figure 7: Nodes Expanded Ratio vs Stranded Persons to Available Locations Ratio for Different Test Cases

Ratio	**Nodes Expanded (Number of Stranded Persons)**			
	Test Case 1	**Test Case 2**	**Test Case 3**	**Test Case 4**
0.25	34 (3)	80 (5)	277 (7)	20 (2)
0.5	99 (7)	353 (11)	695 (14)	41 (4)
0.75	138 (10)	594 (16)	1368 (20)	51 (5)
1.0	179 (13)	912 (21)	1747 (27)	73 (7)
1.5	290 (20)	1621 (32)	2703 (41)	116 (10)

Table 3: Nodes Expanded at Various Ratios of Stranded Persons to Available Locations

Similarly, Figure 7 and Table 3 illustrate that the number of nodes expanded increases substantially as the ratio grows. Test Case 3 consistently results in the highest number of expanded nodes, reflecting its larger grid size and higher density of stranded persons. These findings suggest that more computational resources are required by the planner to find a solution in denser and larger environments.

Overall, the analysis reveals that increasing the ratio of stranded persons to available locations leads to a significant increase in both plan length and the number of nodes expanded. This effect is especially pronounced in larger grids (Test Case 3), where both metrics grow substantially with higher ratios. For smaller grids (Test Cases 1 and 4), the increase in complexity is less dramatic

but still evident. These results indicate that the planner's performance scales with the density of stranded persons, particularly in more complex and larger grid environments.

6 Conclusions

In this project, we implemented and evaluated a rescue drone system using PDDL and the Metric-FF planner, utilizing numeric fluents and types to model complex rescue scenarios. Through a series of test cases with varying grid sizes, numbers of stranded persons, and obstacle configurations, we assessed the planner's performance based on key metrics such as plan length and nodes expanded.

Our analysis reveals that the solution complexity is significantly influenced by the ratio of stranded persons to available locations. As illustrated in the graphs and tables, increasing this ratio leads to a marked rise in both plan length and nodes expanded, particularly evident in larger grids like Test Case 3. This indicates that the planner's computational demands escalate with higher densities of stranded persons, highlighting scalability challenges in more complex environments.

This study also highlights the importance of an efficient problem formulation. Factors such as obstacle placement and grid complexity significantly impact the planner's ability to find optimal rescue paths. More complex environments require greater computational resources, suggesting potential areas for further optimization.

We originally aimed to perform a comparative study between Metric-FF and other planners. However, due to the limited availability of planners that support both numeric fluents and types, we were unable to carry out this experiment. Future work could explore additional planners to assess their performance against Metric-FF.

Overall, this project demonstrates the effectiveness of Metric-FF in handling complex rescue scenarios while highlighting scalability challenges. Future work could explore alternative planners, heuristic strategies, or the influence of obstacle density on solution complexity, especially in larger and more constrained environments.