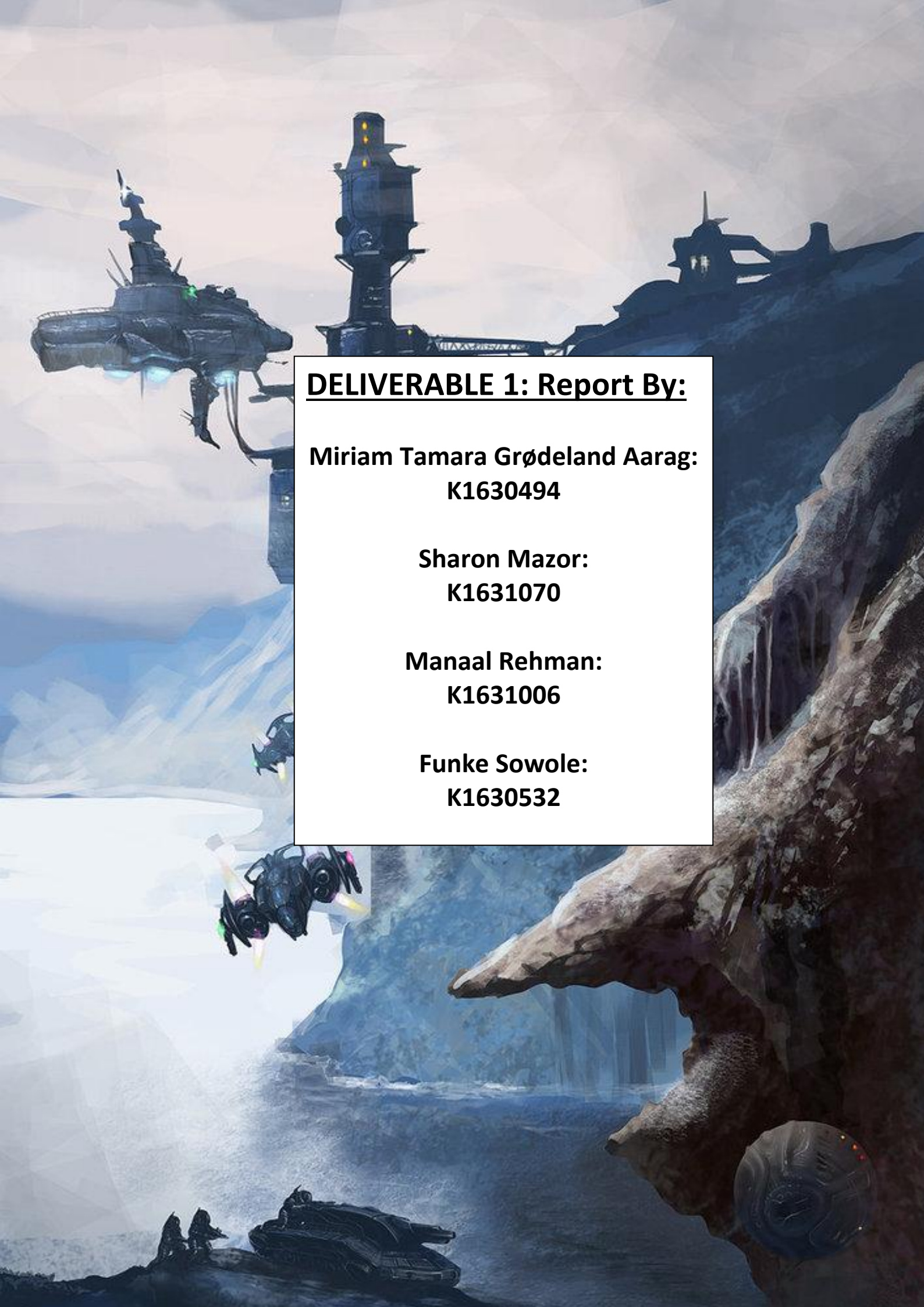


DELIVERABLE 1: Report By:

Miriam Tamara Grødeland Aarag:
K1630494

Sharon Mazor:
K1631070

Manaal Rehman:
K1631006

Funke Sowole:
K1630532

Deliverable 1: Planning Model

Introduction

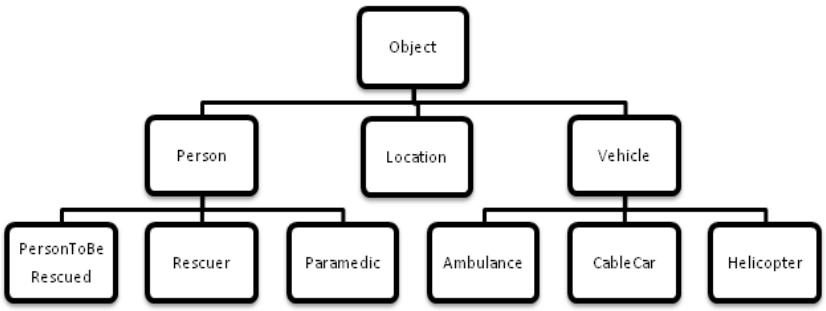
On average 211,000 people in the UK go hiking every year. Of these 9.5% reported injuries and 15% of these injuries were fatal. According to the Swiss Alpine club, the number of deaths due to mountain sport rose by 48% in 2016, and is at its highest since 2011. It is critical that rescue teams reach mountaineers within a reasonable time limits, and safely transport patients to the nearest medical centre.

1) Coming up with an interesting planning domain and instances.

Our planner simulates a Flight & Rescue simulator. We would simulate mountain hikers who need to be rescued from a mountain. This scenario works well as minimising the rescue time is key. An effective planner would save both lives and money. This scenario allows us to experiment with a rich domain, as well as durative actions.

2) Modelling the domain and the instances in PDDL

Our domain consists of:



Our Types Include:

- Person: PersonToBeRescued (the hikers), Rescuer (finds the injured person), Paramedic (treats the injured person)
- Location: There are many locations from where to pick up the injured.
- Vehicle: Ambulance, CableCar, Helicopter

We use the following Predicates:

- (**hasPetrolStation ?l - location**) - if the Vehicle can refuel at this location
- (**personAt ?p - person ?l - location**) - if a person is at a location
- (**vehicleAt ?v - vehicle ?l - location**) - if a vehicle is at a location
- (**injured ?p - personToBeRescued**) - if the hiker is injured
- **(notInjured ?p - personToBeRescued) -** if the hiker is not injured
- **(conscious ?p - personToBeRescued) -** if the hiker is conscious.
- **(notConscious ?p - personToBeRescued) -** if the hiker is unconscious
- **(alive ?p - personToBeRescued) -** if the hiker is alive
- **(onVehicle ?p - person ?v - vehicle) -** if person is onboard vehicle.
- **(withR ?p - personToBeRescued ?r - rescuer) -** if the injured hiker has been found by the rescuer (if the hiker is with the rescuer)
- **(validForAmb ?l - location) -** limits where the ambulance can arrive. (i.e. an ambulance cannot arrive at the top of the mountain)
- **(validForHelicopter ?l - location) -** limits where the helicopter can arrive. (i.e. a helicopter may not be permitted to fly too close to the ground)

- **(validForCC ?l - location) -** limits where the cable car can arrive.
- **(checked ?p - personToBeRescued) -** if the hiker has been checked by a paramedic.

Our Domain's Functions Include:

- **(passengers ?v - vehicle) -** the current number of passengers on the vehicle. This function is connected to durative actions number 1, 3, 4, and 6.
- **(capacity ?v - vehicle) -** the maximum seating capacity for each vehicle. A smaller seating capacity would increase the rescue time. This function is connected to durative actions number 3, 4, and 6.
- **(distance ?from - location ?to - location) -** measure of distance. This function is connected to durative actions number 9, and 12. (The cable car moves between locations, not distances).
- **(withRescuer ?res - rescuer) -** the current number of hikers with the rescuer. Function is connected to durative actions number 2, 7 and 8.
- **(maxWithRescuer ?res - rescuer) -** the maximum number of hikers who can be with the rescuer. Function is connected to durative actions number 7, and 8.
- **(fuelLeft ?v - vehicle) -** the amount of fuel left inside the vehicle. This function is connected to durative actions number 9, 12, 13, 14.
- **(fuelMax ?v - vehicle) -** the vehicle's fuel capacity . A smaller fuel capacity would waste time re-fuelling. This function is connected to durative actions number 13 and 14. The cable car does not use fuel as it it purely electric.
- **(timeForVehicle ?v - vehicle ?from - location ?to - location) -** the time it takes for a vehicle to travel between locations. Some vehicles reach different locations in a faster time. However, some vehicles cannot access certain locations. This function is connected to durative actions number 9, 10, 11 and 12.
- **(timeForRescuer ?r - rescuer ?from - location ?to - location) -** the time the rescuer travels between locations. This function is connected to durative actions number 10.

Our Durative Actions Include:

All of our durative actions assume that the person or vehicle's location match. An example of an durative action is hikers leaving the vehicle, to arrive at their desired location.

1. **disembarkFromVehicle** - All hikers leave the vehicle, at a given location. This action takes 5 minutes. The vehicle must be at the location, and the hiker on the vehicle.
2. **disembarkFromRescuer** - All hikers leave their rescuer, at a given location. This action takes 4 minutes.
3. **boardHelicopter** - The hiker boards the helicopter, at a given location. This action takes 5 minutes. This action can only occur if the hiker is unconscious. Should the helicopter reach its capacity it must undertake multiple journeys.
4. **boardAmbulance** - The hiker boards the ambulance, at a given location. This action takes 5 minutes. The boarding can only occur if the hiker has been checked, and the ambulance has not exceeded its capacity. We could imagine an ambulance full of hikers, and no injured hikers left at location at the end.
5. **checkPerson** - The hiker is checked by the paramedic, at a given location. This action takes 10 minutes.
6. **boardCableCar** - The hiker boards the cable car, at a given location. This action takes 2 minutes. The hiker cannot board the cable car if they are injured, unconscious or dead. We can imagine that at the end we have a cable car full of fit hikers.
7. **secureUninjuredToRescuer** - Any uninjured hikers are secured by a rescuer at a given location. The action takes 3 minutes. The person must NOT be injured. A separate durative action is used for injured hikers, as securing an injured hiker requires more time.
8. **secureInjuredToRescuer** - Injured hikers are secured by a rescuer at a given location. This action takes 5 minutes.

9. **moveHelicopter** - The helicopter moves between 2 locations. This time is calculated using the 'timeForVehicle' function. The helicopter must have enough fuel to fly. Fuel is consumed per-flight. At the end of this action the helicopter arrives at its desired destination.
10. **moveRescuer** - Same as 'moveHelicopter', except the time taken is calculated using the 'timeForRescuer' function, and the rescuer moves instead.
11. **moveCableCar** - Same as 'moveHelicopter', except the cable car moves instead and no fuel is consumed as the cable car is electric.
12. **moveAmbulance** - Same as 'moveHelicopter', except the ambulance moves instead.
13. **refuelAmbulance** - The ambulance can only refuel at certain locations. It takes 10 minutes to refuel. The location must have a petrol station. The end result is a full tank.
14. **refuelHelicopter** - The helicopter can only refuel at certain locations. It takes 10 minutes to refuel. The location must have petrol station. The end result is a full tank.

3) Writing a report describing the planning model.

Our planner finds a solution which rescues hikers from a mountain in the shortest time possible. Its goal is to move the hikers to a better location based on their injury. The rescue team consist of helicopters, paramedics, cable cars, ambulances, rescuers, mountain locations and the hikers to be rescued. Our planner takes into account that hikers have varying levels of injuries – injured/uninjured, conscious/unconscious, alive/dead. A person may be dead and uninjured if they died from a heart attack but dead and injured if they died from a fall. The planner moves the hikers to different locations, based on their injury.

The steps to achieve this should include:

I. Rescuer(s) leave their initial position. They use the durative action '**moveRescuer**' to meet the hiker.
II. Rescuer(s) finds the hikers in their initial state. The paramedics check the person as defined in '**checkPerson**' durative action.
III. The hikers are attached to the rescuer, defined in the '**secureInjuredToRescuer**' and '**secureUninjuredToRescuer**' functions. An uninjured hiker takes less time to secure (3 mins) than an injured person (5 mins) to secure. We could imagine the rescuer being more careful to avoid pain when trying to save an injured hiker, but taking less time in reducing discomfort for the uninjured as any discomfort will only be minor for them. Fewer people injured reduces the time taken to reach the goal state.
IV. The hiker boards an available vehicle, as defined in the durative action. They could '**boardHelicopter/boardAmbulance/boardCableCar**' as decided by the planner. Our planner would decide which is the best vehicle of transport for the rescue as each vehicle has its advantages and disadvantages. I.e. the helicopter travels faster than other vehicles. However, the vehicles have restrictions on where they can travel to. This is specified in the '**validForXXXX**' predicate. Vehicles also have a capacity as defined in the **(capacity ?v - vehicle)** function. A small capacity would require multiple journeys which negatively impacts the total time. Similarly, the rescuer can only carry a set number of hikers, which impacts the total time.
V. It takes a set amount of time for the vehicles to travel between certain locations. Defined in **timeForVehicle ?v - vehicle ?from - location ?to - location)** function. This is based on the distance between the 2 locations.
VI. The '**disembarkFromVehicle'** and **'disembarkFromRescuer'** durative actions produces the goal state. The hikers leave the vehicle and rescuer, and arrive at their desired location.

The planner would then return the time taken to reach the goal state. The total time taken relies on the values used within the problem file.

Appendix (1) - Domain File.

```
(define (domain mountainRescueDomain)
(:requirements :typing :durative-actions :numeric-fluents)
(:types
person location vehicle - object
personToBeRescued rescuer paramedic - person
ambulance cableCar helicopter - vehicle
)
(:predicates
(hasPetrolStation ?l - location)
(personAt ?p - person ?l - location)
(vehicleAt ?v - vehicle ?l - location)
(injured ?p - personToBeRescued)
(notInjured ?p - personToBeRescued)
(conscious ?p - personToBeRescued)
(notConscious ?p - personToBeRescued)
(onVehicle ?p - person ?v - vehicle)
(alive ?p - personToBeRescued)
(withR ?p - personToBeRescued ?r - rescuer)
(validForAmb ?l - location)
(validForHelicopter ?l - location)
(validForCC ?l - location)
(checked ?p - personToBeRescued)
)
(:functions
(passengers ?v - vehicle) - number
(capacity ?v - vehicle) - number
(distance ?from - location ?to - location) - number
(withRescuer ?res - rescuer) - number
(maxWithRescuer ?res - rescuer) - number
(fuelLeft ?v - vehicle) - number
(fuelMax ?v - vehicle) - number
(timeForVehicle ?v - vehicle ?from - location ?to - location) - number
(timeForRescuer ?r - rescuer ?from - location ?to - location) - number
)
(:durative-action disembarkFromVehicle
:parameters (?p - personToBeRescued ?v  - vehicle ?l - location )
:duration (= ?duration 5)
:condition (and (at start (vehicleAt ?v ?l))
                                    (at start (onVehicle ?p ?v)) )
:effect    (and (at start (decrease (passengers ?v) 1))
                                (at start (not (onVehicle ?p ?v)))
                                (at end (personAt ?p ?l) )
                    )
)
(:durative-action disembarkFromRescuer
:parameters (?p - personToBeRescued ?r - rescuer ?l - location)
:duration (= ?duration 4)
:condition (and  (at start (personAt ?r ?l))
                                    (at start (withR ?p ?r)))
```

```
:effect    (and  (at start (decrease (withRescuer ?r) 1))
                                    (at start (not (withR ?p ?r)))
                                    (at end (personAt ?p ?l)))
)
(:durative-action boardHelicopter
:parameters (?p - personToBeRescued ?h - helicopter ?l - location)
:duration (= ?duration 5)
:condition (and (at start (notConscious ?p))
                                    (at start (personAt ?p ?l))
                                    (at start (vehicleAt ?h ?l))
                                    (at start (< (passengers ?h) (capacity ?h))) )
:effect    (and (at start (increase (passengers ?h) 1))
                                    (at end (onVehicle ?p ?h))
                                    (at start (not (personAt ?p ?l))))
)
(:durative-action boardAmbulance
:parameters (?p - personToBeRescued ?a - ambulance ?l - location)
:duration (= ?duration 5)
:condition (and  (at start (checked ?p))
                                    (at start (personAt ?p ?l))
                                    (at start (vehicleAt ?a ?l))
                                    (at start ( < (passengers ?a) (capacity ?a))) )
:effect    (and  (at start (increase (passengers ?a) 1))
                                    (at end (onVehicle ?p ?a))
                                    (at start (not (personAt ?p ?l))))
)
(:durative-action checkPerson
:parameters (?p - personToBeRescued ?d - paramedic ?l - location)
:duration ( = ?duration 10)
:condition         (and (at start (personAt ?p ?l))
                   (at start (personAt ?d ?l)) )
:effect   (at end (checked ?p))
)
(:durative-action boardCableCar
:parameters (?p - personToBeRescued ?cc - cableCar ?l - location)
:duration (= ?duration 2)
:condition (and (at start (conscious ?p))
                                    (at start (alive ?p))
                                    (at start   (personAt ?p ?l))
                                    (at start (vehicleAt ?cc ?l))
                                    (at start (< (passengers ?cc) (capacity ?cc)))
                                    (at start (notInjured ?p)) )
:effect   (and (at start (increase (passengers ?cc) 1))
                                    (at end (onVehicle ?p ?cc))
                                    (at start (not (personAt ?p ?l)) ))
)
(:durative-action secureUninjuredToRescuer
:parameters (?p - personToBeRescued ?r - rescuer ?l -location)
:duration (= ?duration 3)
:condition (and  (at start (notInjured ?p))
                                    (at start (personAt ?p ?l))
```

```
                                    (at start (personAt ?r ?l))
                                    (at start (< (withRescuer ?r) (maxWithRescuer ?r) )) )
:effect  (and  (at end  (withR ?p ?r))
                              (at start (not (personAt ?p ?l))) )
)
(:durative-action secureInjuredToRescuer
:parameters (?p - personToBeRescued ?r - rescuer ?l - location)
:duration (= ?duration 5)
:condition (and (at start (injured ?p))
                                    (at start (personAt ?p ?l))
                                    (at start (personAt ?r ?l))
                                    (at start (< (withRescuer ?r) (maxWithRescuer ?r) )) )
:effect   (and (at end (withR ?p?r))
                                    (at start (not (personAt ?p ?l))) )
)
(:durative-action moveHelicopter
:parameters (?h - helicopter ?from - location ?to - location)
:duration (= ?duration (timeForVehicle ?h ?from ?to))
:condition (and (at start (vehicleAt ?h ?from))
                                    (at start (validForHelicopter ?to))
                                    (at start (>= (fuelLeft ?h) (distance?from ?to) )) )
:effect   (and (at end (vehicleAt ?h ?to))
                                    (at start (not (vehicleAt ?h ?from)))
                                    (at start (decrease (fuelLeft ?h) (distance?from ?to))) )
)
(:durative-action moveRescuer
:parameters (?r - rescuer ?from - location ?to - location)
:duration (= ?duration (timeForRescuer ?r ?from ?to) )
:condition (and  (at start (personAt ?r ?from)) )
:effect    (and (at end (personAt ?r ?to))
                                    (at start (not (personAt ?r ?from))) )
)
(:durative-action moveCableCar
:parameters (?cc - cableCar ?from - location ?to - location)
:duration (= ?duration (timeForVehicle ?cc ?from ?to))
:condition (and  (at start (vehicleAt ?cc ?from))
                                    (at start (validForCC ?to)) )
:effect   (and   (at end (vehicleAt ?cc ?to))
                                    (at start (not (vehicleAt ?cc ?from))) )
)
(:durative-action moveAmbulance
:parameters (?a - ambulance ?from - location ?to - location)
:duration (= ?duration (timeForVehicle ?a ?from ?to))
:condition (and  (at start (vehicleAt ?a ?from))
                    (at start (validForAmb ?to))
                    (at start (>= (fuelLeft ?a) (distance?from ?to) )) )
:effect (and        (at end (vehicleAt ?a ?to))
                    (at start (not (vehicleAt ?a ?from)))
                    (at start (decrease (fuelLeft ?a) (distance?from ?to))) )
)
(:durative-action refuelAmbulance
```

```
:parameters (?a - ambulance ?l - location)
:duration (= ?duration 10)
:condition (and (at start (vehicleAt ?a ?l))
                (over all (vehicleAt ?a ?l))
                (at start (hasPetrolStation ?l)) )
:effect         (at end (assign (fuelLeft ?a) (fuelMax ?a)) )
)
(:durative-action refuelHelicopter
:parameters (?h - helicopter ?l  - location)
:duration (= ?duration 10)
:condition (and (at start (vehicleAt ?h ?l))
        (over all (vehicleAt ?h ?l))
        (at start (hasPetrolStation ?l)) )
:effect (at end (assign (fuelLeft ?h) (fuelMax ?h)) )
)
)
```