## 4CCS1IAI Introduction to Artificial Intelligence Planning Coursework – Deliverable 1

**Group & Members**

|  |  |
| --- | --- |
| Number of Members in the Group: 4 | |
| **Name** | **Kings ID** |
| Mohamed Aflal Mohamed Asker Mohamed Jafferdeen | 1641631 |
| Baha-al-din Aboukoura | 1630516 |
| Muhammed Alhabib Hasan | 1629363 |
| Jaman Salique | 1625442 |

# **Introduction**

London has one of the busiest Ambulance Services in the world, it is in fact the busiest in the United Kingdom. They provide service to more than 8.6 million people who live and work in London. Once a 999 call has been received, the issue is either resolved through the telephone or one of the relevant support ambulance is sent to the patients address. There are many types of vehicles operating within the ambulance service such as trucks and helicopters as well as cars and cycles. Each type of vehicle is dispatched depending on the urgency level of a given patient.

The pressure on the ambulance service has increased very much, such that the London Ambulance Service (LAS) is introducing around 200 more ambulanced to its fleets this year (2017). The raising demand and the complexity of managing the Ambulances efficiently is what gave the inspiration to develop a planning application for LAS -to lay the foundations for an application which could be used to help the ambulance service operate efficiently and effectively.

# **Description**

The purpose behind the application is to allow the planner to come up with the best possible set of actions to reach the goal in the quickest possible way i.e. get the patients to a hospital whilst ensuring the needs of the patients are met such as having the correct transport and being stable before being moved to a vehicle etc.

At present, there are three main objects in the domain, and that is vehicle, patient, and hospital. An ambulance can travel from a hospital to the patient’s location, then stabilise the patients and take the patient to the hospital. The object patient can be in different locations. There can be two types of patients, an urgent patient and a less serious patient, the urgent patient has an urgency of 5 or more, and will receive a helicopter ambulance for stabilising the patient and for taking the patient to the hospital. The less urgent patients will be stabilised and taken to the hospital by land based ambulances. Finally, a hospital can have ambulances and patients, ambulances are dispatched from the location of the hospital.

Our domain consists of the following predicates:

* at-vehicle: used to define the location of the vehicle.
* at-patient: used to define the location of a patient
* patient-stable: is used to set a patient to being stable or not.
* patient-loaded: is used to check if the patient is loaded into the ambulance or not.

* Hospital: used to specify the location of the hospital, so that the ambulance can be directed to the hospital.
* patient-at-hospital: used to make a patient to enter the hospital and to leave the hospital.
* location-linked: is used to link two locations that is in the state space.
* vehicle-full: is a predicate that is used to specify if the ambulance is full.
* vehicle-empty: is used to specify that the vehicle is empty
* fuel-pump: used to specify the location of the fuel pump
* vehicle-prepared: used to specify whether a vehicle is prepared or not.

The domain also contains the following functions:

* fuel-level: used to indicate the level of fuel a vehicle has.
* distance-travelled: holds the total distance travelled by all vehicles during the execution of the plan
* max-fuel: used to specify the max full level of the vehicles fuel tank
* fuel-use: is used to show the fuel used between two locations x and y.
* patient-urgency: is used to give the patient an urgency level. For an example – if the patient has an urgency level of 5 or above, then a helicopter ambulance will be dispatched, and if the urgency level is less than 5, then a land based ambulance will be dispatched.
* vehicle-urgency: used to specify the urgency level the vehicle is used for.
* time-to-prepare: used to set the time taken to prepare a vehicle for use.
* vehicle-speed-up: used to model how much quicker a given vehicle is compared to the standard truck.

The domain also contains the following actions\*:

* drive-to-patient: this actions is used to travel between two locations towards a given patient. It takes a vehicle, from location and to location and a patient as parameters. The location of the vehicle, the fuel level, the number of occupants as well as the urgency of the patient is checked before the action is performed.
* fly-to-patient: this action is the same as the drive-to-patient action with only the duration of the actions differing. The fly action is quicker than the drive action.
* take-patient-to-hospital: this action is used to transport the patient to the hospital. The duration of the action is dependent on the vehicle used. It also takes the current and the destination locations as parameters as well as the vehicle being used and the patient being transported.
* drop-off-patient: this action is used to drop a patient off to the hospital. It has a fixed duration of 3 units of time and takes the location, the patient and the vehicle as parameters.
* stabilize-patient-and-pick-up: this action is used to prepare a patient for transport to the hospital. It takes a fixed duration of 10 units of time. Preconditions of the action include checking that the vehicle doesn’t already contain a patient – among many others.
* refuel-vehicle: this action is used to refuel a vehicle to the maximum fuel level it can have. It takes 10 units of time to carry out.
* prepare-vehicle: this action is used to prepare a vehicle for use. It takes a variable amount of time depending on the vehicle being prepared.

\**for the sake of brevity, the description of actions has been limited. Please refer to the appendix for full details.*

# **Expansions**

As the domain, we are trying to model is quite complex and we have limited time, there are some features which we were unable to add to the domain such as processes. This is something that was introduced in PDDL+ and would require us to use another planner such as DiNo (we are currently using optic clp). Having processes within the domain would allow us to model patient health which would decrease over time. The planner would then aim to minimize the total health lost.

## 4CCS1IAI Introduction to Artificial Intelligence Planning Coursework – Deliverable 2

## **Problem file testing**

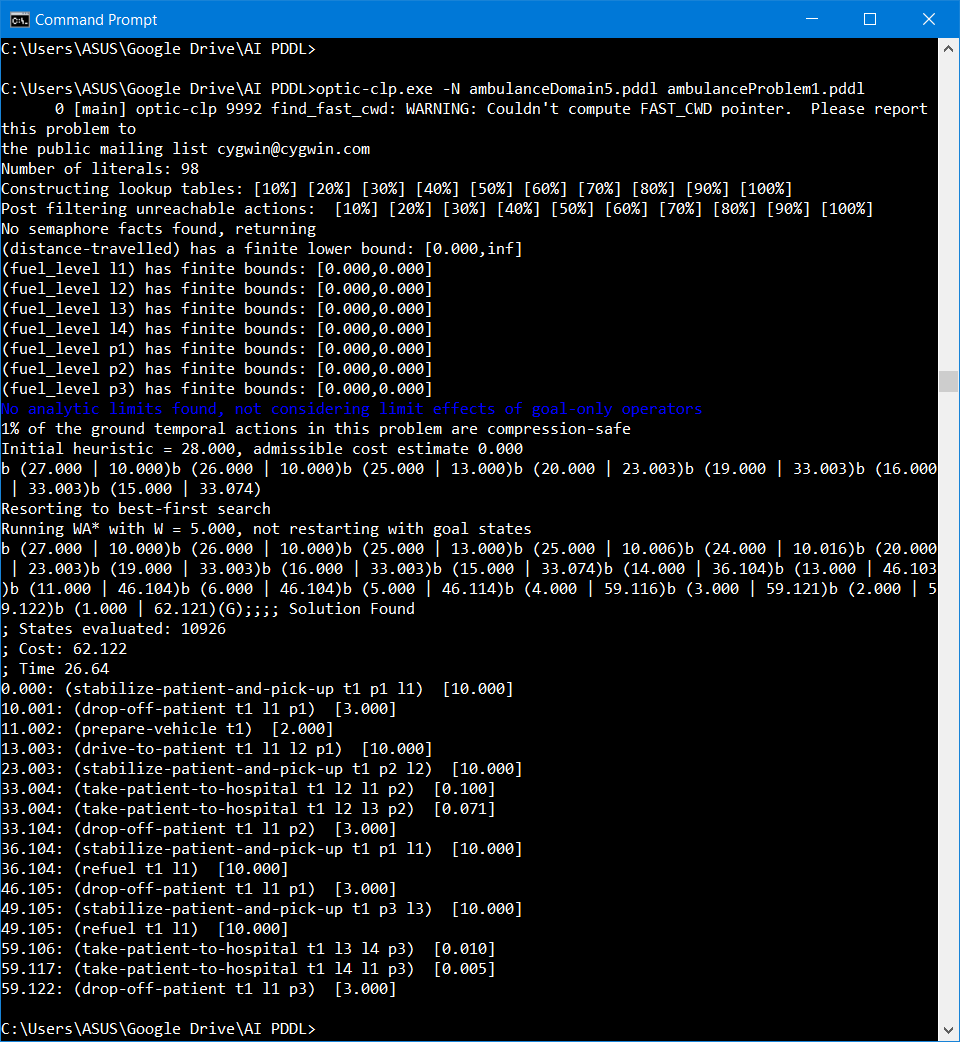
The planner that we used to solve the problems was Optic. The test we conducted took into consideration the first plan, which was not optimised therefore contained unnecessary actions. The reason behind this is that although we would be able to get the most optimised plan for the first few problems, as we scale up it was difficult to even get the un-optimised plan not to mention the optimised plans.

The first problem file (see appendix) that we attempted to solve had the following results:

* No of trucks: 1 - No of helicopters: 0 - No of patients: 3 - No of linked locations: 4 – No of hospitals: 1

All patients were of low urgency

* number of literals: 98

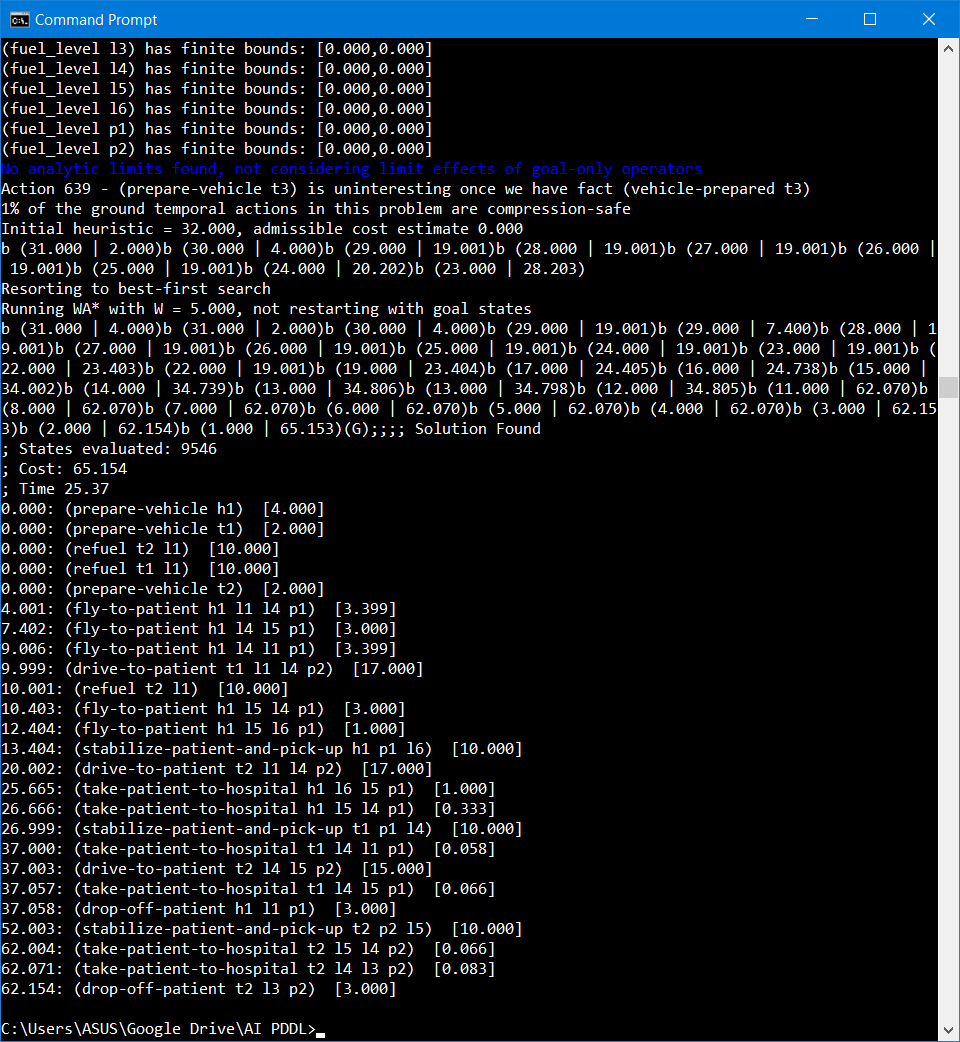


The second problem file (see appendix) that we attempted to solve had the following results:

* No of trucks: 3 - No of helicopters: 1 - No of patients: 2 - No of linked locations: 6 – No of hospitals: 2

One patient was of high priority and the other was of low priority

* number of literals: 147

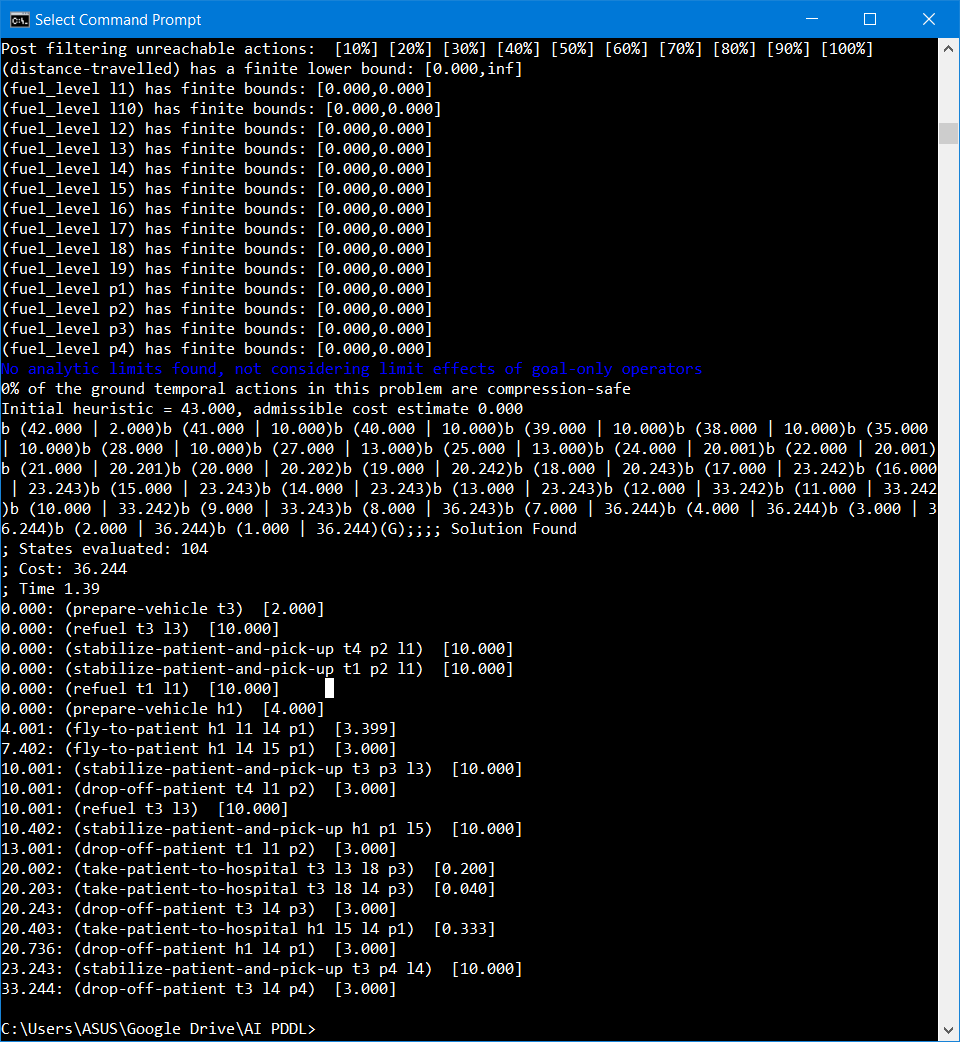


The third problem file (see appendix) that we attempted to solve had the following results:

* No of trucks: 5 - No of helicopters: 1 - No of patients: 4 - No of linked locations: 10 – No of hospitals: 2

One patient was of high priority and the other was of low priority

* number of literals: 435

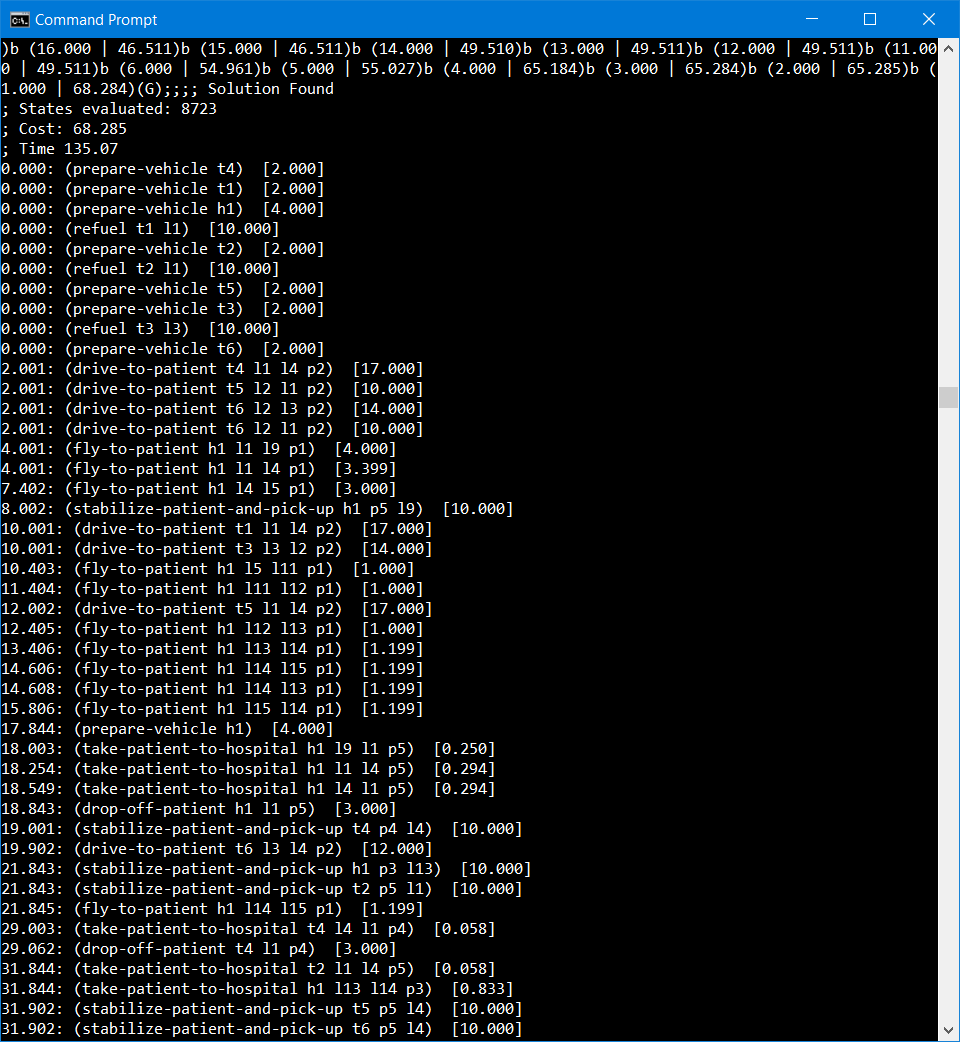


The fourth file (see appendix) that we attempted to solve had the following results:

* No of trucks: 10 - No of helicopters: 1 - No of patients: 5 - No of linked locations: 15 – No of hospitals: 2

Three patients were of high priority and the other two were of low priority

* number of literals: 796



Tests were running on a machine with an Intel Core i5 @ 2.6GHz processor with 8GB of RAM

The overall time to execute a plan seemed to be dependent on the number of literals combined with the number of states needed to evaluate. So, the higher the number of literals and the number of states evaluated the longer the plan took to come up with.

|  |  |  |  |
| --- | --- | --- | --- |
| Problem number | Time | Literals | States Evaluated |
| 1 | 26.64 | 98 | 10926 |
| 2 | 25.37 | 147 | 9546 |
| 3 | 1.39 | 435 | 104 |
| 4 | 135.07 | 796 | 8723 |

From the above table, we can clearly see that the plan that took the longest to compute was the plan with the highest number of literals and a high number of states needed to be evaluated. Whereas the lowest time was for the problem with a high number of literals but a low number of states needed to be evaluated.

# **Scalability**

To see how the domain scales up with increasing numbers of patients, we ran tests by gradually increasing the number of patients to be taken to hospital, below we have documented our findings:

* No of trucks: 6 - No of helicopters: 1 - No of patients: 1 - N - No of linked locations: 15 – No of hospitals: 2

Where N is the value where the machine running the test runs out of ram during the test (i.e. throws a bad allocation exception)

In the test of scalability carried out above, the program threw a bad allocation exception after we had added a sixth patient. This suggests that the domain doesn’t scale very well as it fails to handle more than five patients in an efficient manner. Perhaps this performance issue is related to the machine the program was running on however, I doubt this is the issue as the time gap between 4 and 5 patients show an exponential increase in time after adding 4+ patients. This would suggest that the complexity of the problem increases after adding a sixth patient such that the hardware requirement increases significantly.

Another scalability experiment we conducted included increasing the number of trucks within the domain. Below are our findings:

* No of trucks: 6 - N - No of helicopters: 1 - No of patients: 5 - No of linked locations: 15 – No of hospitals: 2

Where N is the value where the machine running the test runs out of ram during the test (i.e. throws a bad allocation exception)

Strangely during the scalability testing of the number of trucks, the planner failed to successfully create a plan for the domain containing 8 or 9 trucks but successfully managed to create a plan for 10 trucks but not greater than 10. In both cases when testing 8 and 9 trucks, I let the program run until a bad allocation exception was thrown. My guess is that this is down to hardware constraints but I am not fully sure on this as we didn’t get an exception with 10 trucks.

From the experimental analysis and by solving multiple problems it seems that as we increase the number of literals in the problem, the time taken – in very few cases increases at an acceptable level. Adding more and more objects (e.g. patients, vehicles, locations) to the problem does cause some anomalous results such as that seen when adding more trucks and patients to the problem. To make the domain as realistic as possible we would ideally need to be able to handle a significantly larger number of vehicles and patients but as we can see this will come with a dramatic increase in the time and resources required.

In conclusion, it seems that the planner when using the domain and various problem files, can cope with only a small domain i.e. with few patients and few vehicles. In the case where the planner is given a large problem file, the planner requires a machine with higher specifications to successfully produce a plan. The planner also produces many unnecessary steps with each plan – however, these steps are removed once the planner has the chance to optimize the solutions, but again – this optimization demands hardware with a higher specification.

|  |
| --- |
| **Appendix** |

Domain:

(define (domain Ambulance)

(:requirements :strips :typing :fluents :durative-actions:equality)

(:types patient location

helicopter truck - vehicle)

(:predicates

(at-vehicle ?t - vehicle ?l - location)

(patient-stable ?p - patient)

(at-patient ?p - patient ?l - location)

(patient-loaded ?p - patient ?v - vehicle)

(hospital ?h - location)

(patient-at-hospital ?p - patient)

(location-linked ?x ?y - location)

(vehicle-full ?v - vehicle)

(vehicle-empty ?v - vehicle)

(fuel\_pump ?l - location)

(vehicle-prepared ?v - vehicle)

)

(:functions (distance-between ?x ?y - location)

(distance-travelled)

(max\_fuel ?v - vehicle)

(fuel\_level ?v - vehicle)

(fuel\_use ?x ?y - location)

(patient-urgency ?p - patient)

(vehicle-urgency ?v - vehicle)

(time-to-prepare ?v - vehicle)

(vehicle-speed-up ?v - vehicle)

)

(:durative-action drive-to-patient

:parameters (?t - vehicle ?from ?to - location ?p - patient)

:duration (= ?duration (distance-between ?from ?to))

:condition (and (over all (location-linked ?from ?to))

(at start (at-vehicle ?t ?from))

(over all (vehicle-empty ?t))

(at start (vehicle-prepared ?t))

(at start (< (patient-urgency ?p) 5))

(at start (>= (fuel\_level ?t) (fuel\_use ?from ?to))))

:effect (and (at end (not (at-vehicle ?t ?from)))

(at end (at-vehicle ?t ?to))

(at end (increase (distance-travelled) (distance-between ?from ?to)))

(at end (decrease (fuel\_level ?t) (fuel\_use ?from ?to)))

)

)

(:durative-action fly-to-patient

:parameters (?hl - helicopter ?from ?to - location ?p - patient)

:duration (= ?duration (/(distance-between ?from ?to) (vehicle-speed-up ?hl)))

:condition (and (over all (location-linked ?from ?to))

(at start (at-vehicle ?hl ?from))

(at start (vehicle-prepared ?hl))

(at start (>= (patient-urgency ?p)5))

(over all (vehicle-empty ?hl))

(at start (>= (fuel\_level ?hl) (fuel\_use ?from ?to))))

:effect (and (at end (not (at-vehicle ?hl ?from)))

(at end (at-vehicle ?hl ?to))

(at end (increase (distance-travelled) (distance-between ?from ?to)))

(at end (decrease (fuel\_level ?hl) (fuel\_use ?from ?to)))

)

)

(:durative-action take-patient-to-hospital

:parameters (?t - vehicle ?from ?to - location ?p - patient)

:duration (= ?duration (/ (vehicle-speed-up ?t) (distance-between ?from ?to)))

:condition (and (over all (location-linked ?from ?to))

(at start (at-vehicle ?t ?from))

(at start (at-patient ?p ?from))

(at start (patient-stable ?p))

(at start (>= (fuel\_level ?t) (fuel\_use ?from ?to)))

(over all (patient-loaded ?p ?t)))

:effect (and (at end (not (at-vehicle ?t ?from)))

(at end (not (at-patient ?p ?from)))

(at end (at-patient ?p ?to))

(at end (at-vehicle ?t ?to))

(at end (decrease (fuel\_level ?t) (fuel\_use ?from ?to)))

(at end (increase (distance-travelled) (distance-between ?from ?to)))

))

(:durative-action drop-off-patient

:parameters (?t - vehicle ?h - location ?p - patient)

:duration (= ?duration 3)

:condition (and (over all (hospital ?h))

(over all (at-vehicle ?t ?h))

(over all (at-patient ?p ?h))

(at start (patient-loaded ?p ?t)))

:effect (and (at end (patient-at-hospital ?p))

(at end (at-patient ?p ?h))

(at end (not (patient-loaded ?p ?t)))

(at end (not (vehicle-full ?t)))

(at end (not (vehicle-prepared ?t)))

(at end (vehicle-empty ?t)))

)

(:durative-action stabilize-patient-and-pick-up

:parameters (?t - vehicle ?p - patient ?l - location)

:duration (= ?duration 10)

:condition (and (over all (at-vehicle ?t ?l))

(over all (at-patient ?p ?l))

(over all (vehicle-empty ?t))

(over all (>= (patient-urgency ?p) (vehicle-urgency ?t)))

)

:effect (and (at end (patient-stable ?p))

(at end (patient-loaded ?p ?t))

(at end (vehicle-full ?t))

(at end (not (vehicle-empty ?t)))

)

)

(:durative-action refuel

:parameters (?t - vehicle ?l - location)

:duration (= ?duration 10)

:condition (and (at start (fuel\_pump ?l))

(over all (at-vehicle ?t ?l))

(over all (vehicle-empty ?t))

(over all (<= (fuel\_level ?t) (max\_fuel ?t)))

)

:effect (and (at end (increase (fuel\_level ?t) 10)))

)

(:durative-action prepare-vehicle

:parameters (?v - vehicle)

:duration (= ?duration (time-to-prepare ?v))

:condition (and)

:effect (and (at end (vehicle-prepared ?v)))

)

)

Problem 1:

(define (problem simpleProblem)

(:domain Ambulance)

(:objects

t1 - truck

p1 p2 p3 - patient

l1 l2 l3 l4 - location)

(:init

(location-linked l1 l2)

(location-linked l2 l1)

(location-linked l2 l3)

(location-linked l3 l2)

(location-linked l3 l4)

(location-linked l4 l3)

(location-linked l4 l1)

(location-linked l1 l4)

(= (distance-between l1 l2) 10)

(= (distance-between l2 l1) 10)

(= (fuel\_use l1 l2) 5)

(= (fuel\_use l2 l1) 5)

(= (distance-between l2 l3) 14)

(= (distance-between l3 l2) 14)

(= (fuel\_use l2 l3) 5)

(= (fuel\_use l3 l2) 5)

(= (distance-between l3 l4) 100)

(= (distance-between l4 l3) 100)

(= (fuel\_use l3 l4) 5)

(= (fuel\_use l4 l3) 5)

(= (distance-between l4 l1) 200)

(= (distance-between l1 l4) 200)

(= (fuel\_use l4 l1) 5)

(= (fuel\_use l1 l4) 5)

(= (distance-travelled) 0)

(= (max\_fuel t1) 100)

(= (fuel\_level t1) 10)

(fuel\_pump l1)

(at-vehicle t1 l1)

(at-patient p1 l1)

(= (patient-urgency p1) 2)

(at-patient p2 l2)

(= (patient-urgency p2) 4)

(at-patient p3 l3)

(= (patient-urgency p3) 4)

(hospital l1)

(vehicle-empty t1)

(= (time-to-prepare t1) 2)

(= (vehicle-speed-up t1) 1)

(= (vehicle-urgency t1) 0)

)

(:goal (and

(patient-at-hospital p1)

(patient-at-hospital p2)

(patient-at-hospital p3)

))

(:metric minimize (total-time))

)

Problem 2:

(define (problem simpleProblem)

(:domain Ambulance)

(:objects

h1 - helicopter

t1 t2 t3 - truck

p1 p2 - patient

l1 l2 l3 l4 l5 l6 - location)

(:init

(location-linked l1 l2)

(location-linked l2 l1)

(location-linked l2 l3)

(location-linked l3 l2)

(location-linked l3 l4)

(location-linked l4 l3)

(location-linked l4 l1)

(location-linked l1 l4)

(location-linked l4 l5)

(location-linked l5 l4)

(location-linked l5 l6)

(location-linked l6 l5)

(= (distance-between l1 l2) 10)

(= (distance-between l2 l1) 10)

(= (fuel\_use l1 l2) 5)

(= (fuel\_use l2 l1) 5)

(= (distance-between l2 l3) 14)

(= (distance-between l3 l2) 14)

(= (fuel\_use l2 l3) 5)

(= (fuel\_use l3 l2) 5)

(= (distance-between l3 l4) 12)

(= (distance-between l4 l3) 12)

(= (fuel\_use l3 l4) 4)

(= (fuel\_use l4 l3) 4)

(= (distance-between l4 l5) 15)

(= (distance-between l5 l4) 15)

(= (fuel\_use l5 l4) 8)

(= (fuel\_use l4 l5) 8)

(= (distance-between l5 l6) 5)

(= (distance-between l6 l5) 5)

(= (fuel\_use l5 l6) 2)

(= (fuel\_use l6 l5) 2)

(= (distance-between l5 l6) 5)

(= (distance-between l5 l6) 5)

(= (fuel\_use l5 l6) 2)

(= (fuel\_use l6 l5) 2)

(= (distance-between l4 l1) 17)

(= (distance-between l1 l4) 17)

(= (fuel\_use l4 l1) 10)

(= (fuel\_use l1 l4) 10)

(= (distance-travelled) 0)

(= (max\_fuel t1) 100)

(= (max\_fuel t2) 100)

(= (max\_fuel t3) 200)

(= (fuel\_level t1) 10)

(= (fuel\_level t2) 10)

(= (fuel\_level t3) 20)

(= (max\_fuel h1) 500)

(= (fuel\_level h1) 50)

(fuel\_pump l1)

(at-vehicle t1 l1)

(at-vehicle t2 l1)

(at-vehicle t3 l3)

(at-vehicle h1 l1)

(= (vehicle-speed-up t1) 1)

(= (vehicle-speed-up t2) 1)

(= (vehicle-speed-up t3) 1)

(= (vehicle-speed-up h1) 5)

(at-patient p1 l6)

(at-patient p2 l5)

(= (patient-urgency p1) 6)

(= (patient-urgency p2) 4)

(hospital l1)

(hospital l3)

(vehicle-empty t1)

(vehicle-empty t2)

(vehicle-empty t3)

(vehicle-empty h1)

(= (time-to-prepare t1) 2)

(= (time-to-prepare t2) 2)

(= (time-to-prepare t3) 2)

(= (time-to-prepare h1) 4)

(= (vehicle-urgency h1) 5)

(= (vehicle-urgency t1) 0)

(= (vehicle-urgency t2) 0)

)

(:goal (and

(patient-at-hospital p1)

(patient-at-hospital p2)

))

(:metric minimize (total-time))

)

Problem 3:

(define (problem simpleProblem)

(:domain Ambulance)

(:objects

h1 - helicopter

t1 t2 t3 t4 t5 - truck

p1 p2 p3 p4 - patient

l1 l2 l3 l4 l5 l6 l7 l8 l9 l10 - location)

(:init

(location-linked l1 l2)

(location-linked l2 l1)

(location-linked l2 l3)

(location-linked l3 l2)

(location-linked l3 l4)

(location-linked l4 l3)

(location-linked l4 l1)

(location-linked l1 l4)

(location-linked l4 l5)

(location-linked l5 l4)

(location-linked l5 l6)

(location-linked l6 l5)

(location-linked l6 l7)

(location-linked l7 l6)

(location-linked l7 l8)

(location-linked l8 l7)

(location-linked l8 l3)

(location-linked l3 l8)

(location-linked l8 l4)

(location-linked l4 l8)

(location-linked l1 l9)

(location-linked l9 l1)

(location-linked l9 l10)

(location-linked l10 l9)

(location-linked l10 l6)

(location-linked l6 l10)

(location-linked l10 l6)

(location-linked l6 l10)

(= (distance-between l1 l2) 6)

(= (distance-between l2 l1) 6)

(= (fuel\_use l1 l9) 6)

(= (fuel\_use l9 l1) 6)

(= (distance-between l1 l9) 20)

(= (distance-between l9 l1) 20)

(= (fuel\_use l1 l9) 6)

(= (fuel\_use l9 l1) 6)

(= (distance-between l9 l10) 10)

(= (distance-between l10 l9) 10)

(= (fuel\_use l9 l10) 15)

(= (fuel\_use l10 l9) 15)

(= (distance-between l4 l5) 15)

(= (distance-between l5 l4) 15)

(= (fuel\_use l5 l4) 8)

(= (fuel\_use l4 l5) 8)

(= (distance-between l5 l6) 5)

(= (distance-between l5 l6) 5)

(= (fuel\_use l5 l6) 2)

(= (fuel\_use l6 l5) 2)

(= (distance-between l5 l6) 5)

(= (distance-between l5 l6) 5)

(= (fuel\_use l5 l6) 2)

(= (fuel\_use l6 l5) 2)

(= (distance-between l3 l8) 5)

(= (distance-between l8 l3) 5)

(= (fuel\_use l3 l8) 2)

(= (fuel\_use l8 l3) 2)

(= (distance-between l8 l4) 25)

(= (distance-between l4 l8) 25)

(= (fuel\_use l8 l4) 15)

(= (fuel\_use l4 l8) 15)

(= (distance-between l8 l7) 15)

(= (distance-between l7 l8) 15)

(= (fuel\_use l8 l4) 15)

(= (fuel\_use l4 l8) 15)

(= (distance-between l7 l6) 10)

(= (distance-between l6 l7) 10)

(= (fuel\_use l7 l6) 10)

(= (fuel\_use l6 l7) 10)

(= (distance-between l7 l6) 7)

(= (distance-between l6 l7) 7)

(= (fuel\_use l7 l6) 7)

(= (fuel\_use l6 l7) 7)

(= (distance-between l6 l10) 13)

(= (distance-between l10 l6) 13)

(= (fuel\_use l6 l10) 10)

(= (fuel\_use l10 l6) 10)

(= (distance-between l4 l1) 17)

(= (distance-between l1 l4) 17)

(= (fuel\_use l1 l4) 10)

(= (fuel\_use l4 l1) 10)

(= (distance-travelled) 0)

(= (max\_fuel t1) 100)

(= (max\_fuel t2) 100)

(= (max\_fuel t3) 100)

(= (max\_fuel t4) 200)

(= (max\_fuel t5) 200)

(= (fuel\_level t1) 1)

(= (fuel\_level t2) 1)

(= (fuel\_level t3) 2)

(= (fuel\_level t4) 100)

(= (fuel\_level t5) 50)

(= (max\_fuel h1) 500)

(= (fuel\_level h1) 250)

(fuel\_pump l1)

(fuel\_pump l3)

(fuel\_pump l9)

(at-vehicle t1 l1)

(at-vehicle t2 l1)

(at-vehicle t3 l3)

(at-vehicle t4 l1)

(at-vehicle t5 l2)

(at-vehicle h1 l1)

(= (vehicle-speed-up t1) 1)

(= (vehicle-speed-up t2) 1)

(= (vehicle-speed-up t3) 1)

(= (vehicle-speed-up t4) 1)

(= (vehicle-speed-up t5) 1)

(= (vehicle-speed-up h1) 5)

(at-patient p1 l5)

(at-patient p2 l1)

(at-patient p3 l3)

(at-patient p4 l4)

(= (patient-urgency p1) 9)

(= (patient-urgency p2) 4)

(= (patient-urgency p4) 4)

(= (patient-urgency p3) 9)

(hospital l1)

(hospital l4)

(vehicle-empty t1)

(vehicle-empty t2)

(vehicle-empty t3)

(vehicle-empty t4)

(vehicle-empty t5)

(vehicle-empty h1)

(= (time-to-prepare t1) 2)

(= (time-to-prepare t2) 2)

(= (time-to-prepare t3) 2)

(= (time-to-prepare t4) 2)

(= (time-to-prepare t5) 2)

(= (time-to-prepare h1) 4)

(= (vehicle-urgency h1) 5)

(= (vehicle-urgency t1) 0)

(= (vehicle-urgency t2) 0)

(= (vehicle-urgency t3) 0)

(= (vehicle-urgency t4) 0)

(= (vehicle-urgency t5) 0)

)

(:goal (and

(patient-at-hospital p1)

(patient-at-hospital p2)

(patient-at-hospital p3)

(patient-at-hospital p4)

))

(:metric minimize (total-time))

)

Problem 4:

(define (problem simpleProblem)

(:domain Ambulance)

(:objects

h1 - helicopter

t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 - truck

p1 p2 p3 p4 p5 - patient

l1 l2 l3 l4 l5 l6 l7 l8 l9 l10 l11 l12 l13 l14 l15 - location)

(:init

(location-linked l1 l2)

(location-linked l2 l1)

(location-linked l2 l3)

(location-linked l3 l2)

(location-linked l3 l4)

(location-linked l4 l3)

(location-linked l4 l1)

(location-linked l1 l4)

(location-linked l4 l5)

(location-linked l5 l4)

(location-linked l5 l6)

(location-linked l6 l5)

(location-linked l6 l7)

(location-linked l7 l6)

(location-linked l7 l8)

(location-linked l8 l7)

(location-linked l8 l3)

(location-linked l3 l8)

(location-linked l8 l4)

(location-linked l4 l8)

(location-linked l1 l9)

(location-linked l9 l1)

(location-linked l9 l10)

(location-linked l10 l9)

(location-linked l10 l6)

(location-linked l6 l10)

(location-linked l5 l11)

(location-linked l11 l5)

(location-linked l13 l6)

(location-linked l6 l13)

(location-linked l11 l12)

(location-linked l12 l11)

(location-linked l13 l2)

(location-linked l12 l13)

(location-linked l10 l6)

(location-linked l6 l10)

(location-linked l13 l14)

(location-linked l14 l13)

(location-linked l10 l6)

(location-linked l6 l10)

(location-linked l14 l15)

(location-linked l15 l14)

(= (distance-between l1 l2) 10)

(= (distance-between l2 l1) 10)

(= (fuel\_use l1 l2) 5)

(= (fuel\_use l2 l1) 5)

(= (distance-between l2 l3) 14)

(= (distance-between l3 l2) 14)

(= (fuel\_use l2 l3) 5)

(= (fuel\_use l3 l2) 5)

(= (distance-between l3 l4) 12)

(= (distance-between l4 l3) 12)

(= (fuel\_use l3 l4) 4)

(= (fuel\_use l4 l3) 4)

(= (distance-between l1 l9) 20)

(= (distance-between l9 l1) 20)

(= (fuel\_use l1 l9) 6)

(= (fuel\_use l9 l1) 6)

(= (distance-between l9 l10) 10)

(= (distance-between l10 l9) 10)

(= (fuel\_use l9 l10) 15)

(= (fuel\_use l10 l9) 15)

(= (distance-between l4 l5) 15)

(= (distance-between l5 l4) 15)

(= (fuel\_use l5 l4) 8)

(= (fuel\_use l4 l5) 8)

(= (distance-between l5 l6) 5)

(= (distance-between l5 l6) 5)

(= (fuel\_use l5 l6) 2)

(= (fuel\_use l6 l5) 2)

(= (distance-between l5 l6) 5)

(= (distance-between l5 l6) 5)

(= (fuel\_use l5 l6) 2)

(= (fuel\_use l6 l5) 2)

(= (distance-between l3 l8) 5)

(= (distance-between l8 l3) 5)

(= (fuel\_use l3 l8) 2)

(= (fuel\_use l8 l3) 2)

(= (distance-between l8 l4) 25)

(= (distance-between l4 l8) 25)

(= (fuel\_use l8 l4) 15)

(= (fuel\_use l4 l8) 15)

(= (distance-between l8 l7) 15)

(= (distance-between l7 l8) 15)

(= (fuel\_use l8 l4) 15)

(= (fuel\_use l4 l8) 15)

(= (distance-between l7 l6) 10)

(= (distance-between l6 l7) 10)

(= (fuel\_use l7 l6) 10)

(= (fuel\_use l6 l7) 10)

(= (distance-between l7 l6) 7)

(= (distance-between l6 l7) 7)

(= (fuel\_use l7 l6) 7)

(= (fuel\_use l6 l7) 7)

(= (distance-between l6 l10) 13)

(= (distance-between l10 l6) 13)

(= (fuel\_use l6 l10) 10)

(= (fuel\_use l10 l6) 10)

(= (distance-between l4 l1) 17)

(= (distance-between l1 l4) 17)

(= (fuel\_use l1 l4) 10)

(= (fuel\_use l4 l1) 10)

(= (distance-between l5 l11) 5)

(= (distance-between l11 l5) 5)

(= (fuel\_use l11 l5) 4)

(= (fuel\_use l5 l11) 4)

(= (distance-between l12 l11) 5)

(= (distance-between l11 l12) 5)

(= (fuel\_use l11 l12) 4)

(= (fuel\_use l12 l11) 4)

(= (distance-between l12 l13) 5)

(= (distance-between l13 l12) 5)

(= (fuel\_use l12 l13) 4)

(= (fuel\_use l13 l12) 4)

(= (distance-between l6 l13) 3)

(= (distance-between l13 l6) 3)

(= (fuel\_use l12 l13) 2)

(= (fuel\_use l13 l12) 2)

(= (distance-between l14 l13) 6)

(= (distance-between l13 l14) 6)

(= (fuel\_use l14 l13) 6)

(= (fuel\_use l13 l14) 6)

(= (distance-between l14 l15) 6)

(= (distance-between l15 l14) 6)

(= (fuel\_use l14 l15) 6)

(= (fuel\_use l15 l14) 6)

(= (distance-travelled) 0)

(= (max\_fuel t1) 100)

(= (max\_fuel t2) 100)

(= (max\_fuel t3) 100)

(= (max\_fuel t4) 200)

(= (max\_fuel t5) 200)

(= (max\_fuel t6) 200)

(= (max\_fuel t7) 200)

(= (max\_fuel t8) 200)

(= (max\_fuel t9) 200)

(= (max\_fuel t10) 200)

(= (fuel\_level t1) 1)

(= (fuel\_level t2) 1)

(= (fuel\_level t3) 2)

(= (fuel\_level t4) 100)

(= (fuel\_level t5) 50)

(= (fuel\_level t6) 100)

(= (fuel\_level t7) 100)

(= (fuel\_level t8) 100)

(= (fuel\_level t9) 100)

(= (fuel\_level t10) 100)

(= (max\_fuel h1) 500)

(= (fuel\_level h1) 250)

(fuel\_pump l1)

(fuel\_pump l3)

(fuel\_pump l11)

(at-vehicle t1 l1)

(at-vehicle t2 l1)

(at-vehicle t3 l3)

(at-vehicle t4 l1)

(at-vehicle t5 l2)

(at-vehicle t6 l2)

(at-vehicle t7 l2)

(at-vehicle t8 l1)

(at-vehicle t9 l2)

(at-vehicle t10 l2)

(at-vehicle h1 l1)

(= (vehicle-speed-up t1) 1)

(= (vehicle-speed-up t2) 1)

(= (vehicle-speed-up t3) 1)

(= (vehicle-speed-up t4) 1)

(= (vehicle-speed-up t5) 1)

(= (vehicle-speed-up t6) 1)

(= (vehicle-speed-up t7) 1)

(= (vehicle-speed-up t8) 1)

(= (vehicle-speed-up t9) 1)

(= (vehicle-speed-up t10) 1)

(= (vehicle-speed-up h1) 5)

(at-patient p1 l15)

(at-patient p2 l14)

(at-patient p3 l13)

(at-patient p4 l12)

(at-patient p5 l10)

(= (patient-urgency p1) 9)

(= (patient-urgency p2) 2)

(= (patient-urgency p3) 7)

(= (patient-urgency p4) 2)

(= (patient-urgency p5) 7)

(hospital l1)

(hospital l14)

(vehicle-empty t1)

(vehicle-empty t2)

(vehicle-empty t3)

(vehicle-empty t4)

(vehicle-empty t5)

(vehicle-empty t6)

(vehicle-empty t7)

(vehicle-empty t8)

(vehicle-empty t9)

(vehicle-empty t10)

(vehicle-empty h1)

(= (time-to-prepare t1) 2)

(= (time-to-prepare t2) 2)

(= (time-to-prepare t3) 2)

(= (time-to-prepare t4) 2)

(= (time-to-prepare t5) 2)

(= (time-to-prepare t6) 2)

(= (time-to-prepare t7) 2)

(= (time-to-prepare t8) 2)

(= (time-to-prepare t9) 2)

(= (time-to-prepare t10) 2)

(= (time-to-prepare h1) 4)

(= (vehicle-urgency h1) 5)

(= (vehicle-urgency t1) 0)

(= (vehicle-urgency t2) 0)

(= (vehicle-urgency t3) 0)

(= (vehicle-urgency t4) 0)

(= (vehicle-urgency t5) 0)

(= (vehicle-urgency t6) 0)

(= (vehicle-urgency t7) 0)

(= (vehicle-urgency t8) 0)

(= (vehicle-urgency t9) 0)

(= (vehicle-urgency t10) 0)

)

(:goal (and

(patient-at-hospital p1)

(patient-at-hospital p2)

(patient-at-hospital p3)

(patient-at-hospital p4)

(patient-at-hospital p5)

))

(:metric minimize (total-time))

)