5CCS2INT AI Planning Coursework Report

Volunteer Planner - modelling delivery of necessities to at-risk individuals during COVID-19 using PDDL

Sebatian Tranaeus: K17621677 - 1752251

Zakariah Nuccio: K1891842 - 1830790

Jay Vachhani: K1890728 -1827296

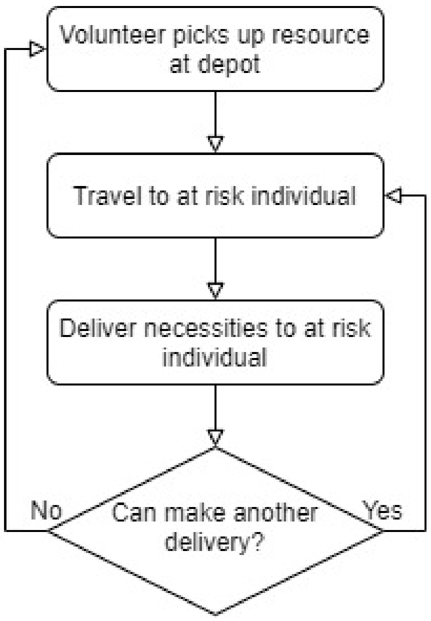
Introduction

With the COVID-19 outbreak, many at-risk individuals need assistance to be able to pull through and stay safe these months. Our domain hopes to model and optimise the operations of community support groups providing necessities such as food and medicine to at-risk individuals that cannot leave their homes.

Specifically, volunteers can pick up necessities from depots(representing community support centres, or stores), and deliver these necessities to at-risk persons in their community. Volunteers can travel by walking, cycling, or driving, and each mode of travel will vary in terms of speed and carrying capacity. Volunteers can work for a specified number of minutes before they take a 10 minute break, after which they can return to work.

Our goal state is that all necessity requests from at-risk persons are satisfied. As a real world situation, this is also the goal of all the frontline volunteers working today to provide help and support to those in need.

This figure below represents the flow of how the different objects should behave when the planner goes through a given problem file. Volunteers often utilise their own modes of transport and so they will use the same mode of transport throughout. They travel to the at risk individual and then deliver necessities. Then, if they have more resources, they will look at the next best at risk individual to reach. Otherwise they will travel back to the depot to collect resources for the next delivery that is not satisfied.


Volunteer picks up resource at depot
Travel to at risk individual
Deliver necessities to at risk individual
Can make another delivery?
No
Yes

We choose to not handle the fuel and the refuelling process of a mode of travel such as a car. This was because the plans generated from this domain would be for a day's work, and a fair assumption of the domain is that all modes of transport are ready for use for the entire day. Again, the focus of the domain is on distribution of resources.

We also choose to not allow multiple volunteers to share a car or other vehicle, given the social distancing rules that they would have to operate under.

Task 1 - Planning Domain

Our domain leverages the following PDDL features: fluents, durative-actions, typing, duration-inequalities, equality

Types

- **locatable** of type object - represents any object that can have a location
- **volunteer**, an instance of locatable - represents a community volunteer
- **atRiskPerson**, an instance of locatable - represents an at risk person that must stay home
- **depot**, an instance of locatable - represents a community centre or store where volunteers can pick up resources
- **modeOfTravel** of type objects - represents a method volunteers use to travel between locations. Eg walking, cycling, driving

- **resource** - represents a necessity that an atRiskPerson can require, or a volunteer deliver. Eg medicine, food

Predicates

- **(at ?obj - locatable ?l - locatable)** - used for representing that an object is at a specific location
- **(linked ?l1 ?l2 - locatable)** - used for representing that two locatable objects are geographically linked
- **(usingTransport ?v - volunteer ?m - modeOfTravel)** - used for representing that a volunteer is using a specific mode of travel while they volunteer
- **(available ?v - volunteer)** - used for representing that a volunteer is not busy and is available to do work. This is to force volunteers to execute only one action at a time.

Functions

- **(time-to-arrive ?from ?to - locatable ?mode - modeOfTravel)** - the time it takes to travel, using the given mode of travel, from and to the given locations. Represented in minutes.
- **(requires ?p - atRiskPerson ?r - resource)** - how many of a given resource the person at risk requires.
- **(resources-stored ?r - resource ?obj - locatable)** - how many of a given resource is stored at an object. For example, this can be used to represent resources stored at a depot, or resources being carried by a volunteer.
- **(capacity ?v -volunteer ?mode - modeOfTravel)** - the capacity of the given volunteer using the given mode of travel to carry resources. This is not necessarily the number of resources a volunteer can carry, but more the amount of space that volunteer has. Represented using litres for volume.
- **(resource-size ?r - resource)** - the size of the given resource. This is what matters when considering the capacity of the volunteer. Represented using litres for volume. For example, a carton of milk might have capacity 1, while a back of antibiotics might have capacity 0.2.
- **(activePeriod ?v - volunteer)** - The number of minutes a volunteer can work before needing a 10 minute rest.
- **(remainingActivePeriod ?v - volunteer)** - The number of minutes a volunteer has left before their shift, of duration activePeriod, is complete and they go on a 10 minute break.

Durative Actions

- **PICK-UP-RESOURCE (?v - volunteer ?m - modeOfTravel ?r - resource ?d - depot)**

  Represents a volunteer, using the given mode of travel, picking up a specific resource from the given depot. The volunteer must be available, at the depot, have enough time left before their break, and be using the given mode of travel. Additionally the volunteer must have capacity for the given resource, and the depot must have at least one instance of that resource stored. The key effects are that the volunteer is not available for the duration of the action, the volunteer starts storing that resource, while the depot stops, the action's duration is deducted from the volunteer's remaining time, and the capacity of the volunteer is reduced by the size of the resource.

- **DELIVER-RESOURCE (?v - volunteer ?m - modeOfTravel ?r - resource ?p - atRiskPerson)**

  Represents a volunteer, using the given mode of travel, delivering a resource to the given atRiskPerson. The volunteer must be available, at the atRiskPerson, have enough time left before their break, and be using the given mode of travel. The key effects are that the volunteer is not available for the duration of the action, the volunteer stops storing that resource, while the atRiskPerson starts, the action's duration is deducted from the volunteer's remaining time, and the capacity of the volunteer is increased by the size of the resource.

- **TRAVEL (?v - volunteer ?from ?to - locatable, ?mode - modeOfTravel)**

  Represents a volunteer travelling from one location to another, using the given mode of travel. Requires that the to and from locations are linked together, that the volunteer is at the from location, has enough time left to travel to the to location, that the volunteer is available, and that it is using the given mode of transport. The effect is that the volunteer is not available for the duration of the action, it has moved to the to location, and that its active remaining time has decreased by the duration of the action.

- **TRANSFER-RESOURCE (?v1 ?v2 - volunteer ?m1 ?m2 - modeOfTravel ?r - resource ?l - locatable)**

  Represents volunteer v1 giving a resource to volunteer v2, essentially an exchange of resources. Requires that v1 has the resource on hand, v2 has the capacity to carry, and that v1 and v2 are in the same location. The idea behind this was allowing the planner to develop its own sub-distribution systems within communities. For example, if one volunteer was in a car, while several others were walking or cycling nearby, they could use the carrying capacity of the car to distribute resources optimally in an area far away from a depot.

- **REST (?v - volunteer)**

  Represents when a volunteer is resting for 10 minutes. We thought accounting for the need of volunteers to take breaks would play a key role in modeling our domain's real world situation. This action increases the volunteer's remainingActivePeriod by their activePeriod value, allowing them to effectively work another shift. The volunteer is not available for the duration of the action.

Appendix - Domain

```
;Volunteer Planner

(define (domain volunteerPlanner)

(:requirements :fluents :durative-actions :typing :duration-inequalities :equality)

(:types
    locatable - object
    volunteer atRiskPerson depot - locatable
    modeOfTravel
    resource
)


(:predicates
    (at ?obj - locatable ?l - locatable)
    (linked ?l1 ?l2 - locatable)
    (usingTransport ?v - volunteer ?m - modeOfTravel)
    (available ?v - volunteer)
)


(:functions
    (time-to-arrive ?from ?to - locatable ?mode - modeOfTravel)
    (requires ?p - atRiskPerson ?r - resource)
    (resources-stored ?r - resource ?obj - locatable)
    (capacity ?v -volunteer ?mode - modeOfTravel)
    (resource-size ?r - resource)
    (activePeriod ?v - volunteer)
    (remainingActivePeriod ?v - volunteer)
)

(:durative-action PICK-UP-RESOURCE
    :parameters (?v - volunteer ?m - modeOfTravel ?r - resource ?d - depot)
    :duration (= ?duration 0.2)
    :condition (and
        (at start (at ?v ?d))
        (at start (> (remainingActivePeriod ?v) 0.2))
        (at start (usingTransport ?v ?m))
        (at start (>= (capacity ?v ?m) (resource-size ?r)))
        (at start (>= (resources-stored ?r ?d) 1))
        (at start (available ?v))
    )
    :effect (and
        (at start (not (available ?v)))
        (at end (decrease (remainingActivePeriod ?v) 0.2))
        (at end (decrease (resources-stored ?r ?d) 1))
        (at end (increase (resources-stored ?r ?v) 1))
        (at end (decrease (capacity ?v ?m) (resource-size ?r)))
        (at end (available ?v))
    )
)

(:durative-action DELIVER-RESOURCE
    :parameters (?v - volunteer ?m - modeOfTravel ?r - resource ?p - atRiskPerson)
    :duration (= ?duration 0.2)
    :condition (and
        (at start (at ?v ?p))
        (at start (> (remainingActivePeriod ?v) 0.2))
        (at start (usingTransport ?v ?m))
        (at start (>= (resources-stored ?r ?v) 1))
        (at start (available ?v))
    )
    :effect (and
        (at start (not (available ?v)))
        (at end (decrease (remainingActivePeriod ?v) 0.2))
        (at end (decrease (resources-stored ?r ?v) 1))
        (at end (decrease (requires ?p ?r) 1))
        (at end (increase (capacity ?v ?m) (resource-size ?r)))
        (at end (available ?v))
    )
)

(:durative-action TRAVEL
    :parameters (?v - volunteer ?from ?to - locatable, ?mode - modeOfTravel)
    :duration (= ?duration (time-to-arrive ?from ?to ?mode))
    :condition (and
        (at start (> (remainingActivePeriod ?v) (time-to-arrive ?from ?to ?mode)))
        (at start (at ?v ?from))
        (at start (usingTransport ?v ?mode))
        (at start (linked ?from ?to))
        (at start (available ?v))
    )
    :effect (and
        (at start (not (available ?v)))
        (at start (decrease (remainingActivePeriod ?v) (time-to-arrive ?from ?to ?mode)))
        (at end (not (at ?v ?from)))
        (at end (at ?v ?to))
        (at end (available ?v))
    )
)

(:durative-action TRANSFER-RESOURCE
    :parameters (?v1 ?v2 - volunteer ?m1 ?m2 - modeOfTravel ?r - resource ?l - locatable)
    :duration (= ?duration 0.4)
    :condition (and
        (at start (available ?v1))
        (at start (> (remainingActivePeriod ?v1) 0.4))
        (at start (available ?v2))
```

```
            (at start (> (remainingActivePeriod ?v2) 0.4))
            (at start (at ?v1 ?l))
            (at start (at ?v2 ?l))
            (at start (usingTransport ?v1 ?m1))
            (at start (usingTransport ?v2 ?m2))
            (at start (>= (resources-stored ?r ?v1) 1))
            (at start (>= (capacity ?v2 ?m2) (resource-size ?r)))
        )
        :effect (and
            (at start (not (available ?v1)))
            (at end (decrease (remainingActivePeriod ?v1) 0.4))
            (at start (not (available ?v2)))
            (at end (decrease (remainingActivePeriod ?v2) 0.4))
            (at end (decrease (resources-stored ?r ?v1) 1))
            (at end (increase (capacity ?v1 ?m1) (resource-size ?r)))
            (at end (increase (resources-stored ?r ?v2) 1))
            (at end (decrease (capacity ?v2 ?m2) (resource-size ?r)))
            (at end (available ?v1))
            (at end (available ?v2))
        )
    )

    (:durative-action REST
        :parameters (?v - volunteer)
        :duration (= ?duration 10)
        :condition (and
            (at start (< (remainingActivePeriod ?v) 3))
            (at start (available ?v))
        )
        :effect (and
            (at start (not (available ?v)))
            (at end (increase (remainingActivePeriod ?v) (activePeriod ?v)))
            (at end (available ?v))
        )
    )
)
```

Task 2 - Solving the Planning Domain

Problems

We chose to use the OPTIC planner for this project. We wrote several different problems to represent different real world situations, with variations including:

- Number of at risk persons
- Different mode of transport configurations
- How clustered groups of at risk persons are. This is to represent situations where at risk persons might live in the same street or apartment block.
- Number of depots
- Number of volunteers

- Number of types of resources to be distributed

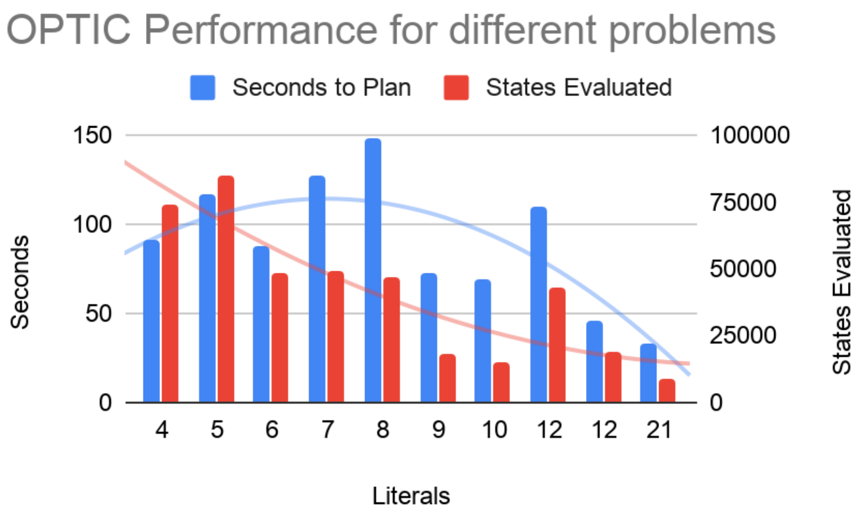
In general, we focused on representing a single local community in each problem. We did not see the utility in trying to solve for multiple communities and areas at the same time, given that, in reality, these communities operate independently.

We also tried to always represent a dynamic where there are significantly more atRiskPersons compared to volunteers, as this is the case in reality.

Analysis

Initially we create a comprehensive range of problems for this analysis. However, the more complicated problems, for example those including lots of interconnected clusters of atRiskPersons, were very quickly killed by the planner when we ran them. With this in mind, and to make sure we could have some amount of data in our graph, we created several more problems that are incrementally larger than our simplest problem, problem1.

The below graph shows specific metrics for the OPTIC planner as it tries to generate plans for problems 1, 8 , 9, 10, 11, 12, 13, and 3. Please note that, if a problem is included in this graph, it is because the planner was able to generate a plan, but not necessarily an optimal one(this would mean the planner killed its operation while planning).



Looking at just the graph would suggest an inverse relationship between the number of literals and the seconds to generate a plan. We believe this is due to the fact that the planner was not able to generate fully optimised plans for the larger, more complex problems. This displays a limitation of our current data.

However, looking at the first 7 problems in the graph suggests the planner was able to handle gradual increases in problem size and complexity. Although clearly, there is a cutoff point to this trend given the inability of the planner to solve more complex problems.

Problems 4, 5, and 7, each with 28, 27, and 12 literals respectively, were killed by the planner before any plan was generated. This is why they are not on the graph, but can still be found in the appendix. All these problems represented more complex situations, such as significantly more atRiskPersons, resources, or depots, or significantly more links between locations. It's clear the planner was struggling to scale with these problems.

We faced several problems while developing this domain:

- Making sure volunteers cannot perform multiple actions at once - this was implemented by using the available predicate to make sure volunteers operate synchronously and actions are performed atomically.
- Representing modes of travel - initially we hardcoded the types of mode of travel in the domain(cycling, walking, etc). However this proved inflexible when it came to varying the problem files, so we decided to move modeOfTravel definition to the problem files, instead of hardcoding it in the domain file.
- For more complicated problems, we found the planner took too long to generate solutions for them, and would sometimes prematurely end its search before it could finish. We found that in some cases, adding the -c option when running OPTIC, which changes how the RPG heuristic prioritises different choices in its graph, would allow it to both find solutions and do so more quickly.
- Making sure volunteers do not find themselves in a situation where their activeRemainingDuration is too low to do anymore actions, but not low enough(eg zero) to allow them to rest - we resolved this by allowing the REST action to be executed if the volunteer has an activeRemainingDuration below three. For future iterations we believe a non-hardcoded solution is optimal.

Overall, we believe we've developed a domain and set of problem files that are representative of the fundamental situation facing community support groups during the Covid-19 crisis in the UK. Future areas of improvement for the domain would be working on scalability, and investigating the behaviour of the REST action(ensuring that no volunteer finds itself in the last problem mentioned above).

Appendix - All Problem Files

Problem 1

```
(define (problem problem1) (:domain volunteerPlanner)
(:objects
    vWilliam - volunteer
    arRaff arBen - atRiskPerson
    dStrand - depot
    egg painkillers - resource
    walking cycling - modeOfTravel
)

(:init
    ; resources
    (= (resource-size egg) 0.2)
    (= (resource-size painkillers) 0.4)

    ; depots
    (= (resources-stored egg dStrand) 100)
    (= (resources-stored painkillers dStrand) 50)

    ; volunteers
    (at vWilliam dStrand)
    (available vWilliam)
    (= (activePeriod vWilliam) 100)
    (= (remainingActivePeriod vWilliam) 100)
    (usingTransport vWilliam walking)
    (= (capacity vWilliam walking) 10)
    (= (resources-stored egg vWilliam) 0)
    (= (resources-stored painkillers vWilliam) 0)

    ; links
    (linked dStrand arRaff)
    (linked arRaff dStrand)
    (= (time-to-arrive dStrand arRaff walking) 10)
    (= (time-to-arrive arRaff dStrand walking) 10)

    (linked dStrand arBen)
    (linked arBen dStrand)
    (= (time-to-arrive dStrand arBen walking) 6)
    (= (time-to-arrive arBen dStrand walking) 6)

    ; requirements
    (= (requires arBen egg) 1)
    (= (requires arBen painkillers) 6)
    (= (requires arRaff egg) 2)
    (= (requires arRaff painkillers) 1)
)

(:goal (and
    (= (requires arBen egg) 0)
    (= (requires arBen painkillers) 0)
    (= (requires arRaff egg) 0)
    (= (requires arRaff painkillers) 0)
))
)
```

Problem 2

```
(define (problem problem2) (:domain volunteerPlanner)
(:objects
    vWilliam, vDan - volunteer
    arRaff arBen arGiulio - atRiskPerson
    dStrand dWaterloo - depot
    egg painkillers - resource
    walking cycling - modeOfTravel
)

(:init
    ; resources
    (= (resource-size egg) 0.2)
    (= (resource-size painkillers) 0.4)

    ; depots
    (= (resources-stored egg dStrand) 100)
    (= (resources-stored painkillers dStrand) 50)
    (= (resources-stored egg dWaterloo) 10)
    (= (resources-stored painkillers dWaterloo) 5)

    ; volunteers
    (at vWilliam dStrand)
    (available vWilliam)
    (usingTransport vWilliam walking)
    (= (activePeriod vWilliam) 60)
    (= (remainingActivePeriod vWilliam) 60)
    (= (capacity vWilliam walking) 10)
    (= (resources-stored egg vWilliam) 0)
    (= (resources-stored painkillers vWilliam) 0)

    (at vDan dStrand)
    (available vDan)
    (usingTransport vDan walking)
    (= (activePeriod vDan) 60)
    (= (remainingActivePeriod vDan) 60)
    (= (capacity vDan walking) 10)
    (= (resources-stored egg vDan) 0)
    (= (resources-stored painkillers vDan) 0)

    ; links
    (linked dStrand arRaff)
    (linked arRaff dStrand)
    (= (time-to-arrive dStrand arRaff walking) 10)
    (= (time-to-arrive arRaff dStrand walking) 10)
    (= (time-to-arrive dStrand arRaff cycling) 3)
    (= (time-to-arrive arRaff dStrand cycling) 3)

    (linked dStrand arBen)
    (linked arBen dStrand)
    (= (time-to-arrive dStrand arBen walking) 6)
    (= (time-to-arrive arBen dStrand walking) 6)
    (= (time-to-arrive dStrand arBen cycling) 2)
    (= (time-to-arrive arBen dStrand cycling) 2)

    (linked dWaterloo arGiulio)
    (linked arGiulio dWaterloo)
    (= (time-to-arrive dWaterloo arGiulio walking) 12)
    (= (time-to-arrive arGiulio dWaterloo walking) 12)
    (= (time-to-arrive dWaterloo arGiulio cycling) 4)
    (= (time-to-arrive arGiulio dWaterloo cycling) 4)

    (linked dWaterloo dStrand)
    (linked dStrand dWaterloo)
    (= (time-to-arrive dWaterloo dStrand walking) 20)
    (= (time-to-arrive dStrand dWaterloo walking) 20)
    (= (time-to-arrive dWaterloo dStrand cycling) 7)
    (= (time-to-arrive dStrand dWaterloo cycling) 7)

    ; requirements
    (= (requires arRaff egg) 2)
    (= (requires arRaff painkillers) 1)
    (= (requires arBen egg) 1)
    (= (requires arBen painkillers) 6)
    (= (requires arGiulio egg) 4)
    (= (requires arGiulio painkillers) 8)
)

(:goal (and
    (= (requires arBen egg) 0)
    (= (requires arBen painkillers) 0)
    (= (requires arRaff egg) 0)
    (= (requires arRaff painkillers) 0)
    (= (requires arGiulio egg) 0)
    (= (requires arGiulio painkillers) 0)
))
)
```

Problem 3

```
(define (problem problem3) (:domain volunteerPlanner)
(:objects
    vWilliam, vDan, vJeff - volunteer
    arRaff arBen arGiulio arJay - atRiskPerson
    dStrand dWaterloo - depot
    egg painkillers - resource
    walking cycling driving - modeOfTravel
)

(:init
    ; resources
    (= (resource-size egg) 0.2)
    (= (resource-size painkillers) 0.4)

    ; depots
    (= (resources-stored egg dStrand) 100)
    (= (resources-stored painkillers dStrand) 50)
    (= (resources-stored egg dWaterloo) 10)
    (= (resources-stored painkillers dWaterloo) 5)


    ; volunteers
    (at vWilliam dStrand)
    (available vWilliam)
    (usingTransport vWilliam walking)
    (= (capacity vWilliam walking) 10)
    (= (activePeriod vWilliam) 60)
    (= (remainingActivePeriod vWilliam) 60)
    (= (resources-stored egg vWilliam) 0)
    (= (resources-stored painkillers vWilliam) 0)

    (at vDan dStrand)
    (available vDan)
    (usingTransport vDan cycling)
    (= (capacity vDan cycling) 6)
    (= (activePeriod vDan) 60)
    (= (remainingActivePeriod vDan) 60)
    (= (resources-stored egg vDan) 0)
    (= (resources-stored painkillers vDan) 0)

    (at vJeff dStrand)
    (available vJeff)
    (usingTransport vJeff driving)
    (= (capacity vJeff driving) 30)
    (= (activePeriod vJeff) 60)
    (= (remainingActivePeriod vJeff) 60)
    (= (resources-stored egg vJeff) 0)
    (= (resources-stored painkillers vJeff) 0)

    ; links
    (linked dStrand arRaff)
    (linked arRaff dStrand)
    (= (time-to-arrive dStrand arRaff walking) 10)
    (= (time-to-arrive arRaff dStrand walking) 10)
    (= (time-to-arrive dStrand arRaff driving) 5)
    (= (time-to-arrive arRaff dStrand driving) 5)
    (= (time-to-arrive dStrand arRaff cycling) 3)
    (= (time-to-arrive arRaff dStrand cycling) 3)

    (linked dStrand arBen)
    (linked arBen dStrand)
    (= (time-to-arrive dStrand arBen walking) 4)
    (= (time-to-arrive arBen dStrand walking) 4)
    (= (time-to-arrive dStrand arBen cycling) 2)
    (= (time-to-arrive arBen dStrand cycling) 2)
    (= (time-to-arrive dStrand arBen driving) 3)
    (= (time-to-arrive arBen dStrand driving) 3)

    (linked dWaterloo arJay)
    (linked arJay dWaterloo)
    (= (time-to-arrive dWaterloo arJay walking) 25)
    (= (time-to-arrive arJay dWaterloo walking) 25)
    (= (time-to-arrive dWaterloo arJay cycling) 20)
    (= (time-to-arrive arJay dWaterloo cycling) 20)
    (= (time-to-arrive dWaterloo arJay driving) 12)
    (= (time-to-arrive arJay dWaterloo driving) 12)

    (linked dStrand arJay)
    (linked arJay dStrand)
    (= (time-to-arrive dStrand arJay walking) 40)
    (= (time-to-arrive arJay dStrand walking) 40)
    (= (time-to-arrive dStrand arJay cycling) 25)
    (= (time-to-arrive arJay dStrand cycling) 25)
    (= (time-to-arrive dStrand arJay driving) 15)
    (= (time-to-arrive arJay dStrand driving) 15)

    (linked dWaterloo arGiulio)
    (linked arGiulio dWaterloo)
    (= (time-to-arrive dWaterloo arGiulio walking) 5)
    (= (time-to-arrive dWaterloo arGiulio walking) 5)
    (= (time-to-arrive arGiulio dWaterloo cycling) 2)
    (= (time-to-arrive arGiulio dWaterloo cycling) 2)
    (= (time-to-arrive dWaterloo arGiulio driving) 2)
    (= (time-to-arrive arGiulio dWaterloo driving) 2)

    (linked dWaterloo dStrand)
    (linked dStrand dWaterloo)
    (= (time-to-arrive dWaterloo dStrand walking) 20)
    (= (time-to-arrive dStrand dWaterloo walking) 20)
    (= (time-to-arrive dWaterloo dStrand cycling) 7)
    (= (time-to-arrive dStrand dWaterloo cycling) 7)
    (= (time-to-arrive dWaterloo dStrand driving) 5)
    (= (time-to-arrive dStrand dWaterloo driving) 5)


    ; requirements
    (= (requires arRaff egg) 2)
    (= (requires arRaff painkillers) 1)
    (= (requires arBen egg) 1)
    (= (requires arBen painkillers) 6)
    (= (requires arGiulio egg) 4)
    (= (requires arGiulio painkillers) 8)
    (= (requires arJay egg) 5)
    (= (requires arJay painkillers) 5)

)

(:goal (and
    (= (requires arBen egg) 0)
    (= (requires arBen painkillers) 0)
    (= (requires arRaff egg) 0)
    (= (requires arRaff painkillers) 0)
    (= (requires arGiulio egg) 0)
    (= (requires arGiulio painkillers) 0)
    (= (requires arJay egg) 0)
    (= (requires arJay painkillers) 0)
))
)
```

Problem 4

```
(define (problem clustersProblem) (:domain volunteerPlanner)
(:objects
    vWilliam, vDan - volunteer
    arRaff arBen arGiulio - atRiskPerson
    arSeb arJeffrey arZak arJay - atRiskPerson
    arKolling arColes arKeppens arAgi - atRiskPerson
    dStrand dWaterloo - depot
    egg painkillers - resource
    walking - modeOfTravel
)

(:init
    ; resources
    (= (resource-size egg) 0.2)
    (= (resource-size painkillers) 0.4)

    ; depots
    (= (resources-stored egg dStrand) 100)
    (= (resources-stored painkillers dStrand) 50)
    (= (resources-stored egg dWaterloo) 10)
    (= (resources-stored painkillers dWaterloo) 5)


    ; volunteers
    (at vWilliam dStrand)
    (available vWilliam)
    (usingTransport vWilliam walking)
    (= (capacity vWilliam walking) 10)
    (= (activePeriod vWilliam) 100)
    (= (remainingActivePeriod vWilliam) 100)
    (= (resources-stored egg vWilliam) 0)
    (= (resources-stored painkillers vWilliam) 0)

    (at vDan dWaterloo)
    (available vDan)
    (usingTransport vDan walking)
    (= (capacity vDan walking) 10)
    (= (activePeriod vDan) 100)
    (= (remainingActivePeriod vDan) 100)
    (= (resources-stored egg vDan) 0)
    (= (resources-stored painkillers vDan) 0)

    ; links
    ; depot links
    (linked dWaterloo dStrand)
    (linked dStrand dWaterloo)
    (= (time-to-arrive dWaterloo dStrand walking) 20)
    (= (time-to-arrive dStrand dWaterloo walking) 20)

    ; cluster 1
    ; cluster 1 requirements
    (= (requires arRaff egg) 2)
    (= (requires arRaff painkillers) 1)
    (= (requires arBen egg) 1)
    (= (requires arBen painkillers) 6)
    (= (requires arGiulio egg) 4)
    (= (requires arGiulio painkillers) 8)

    ; cluster 1 links
    (linked dStrand arRaff)
    (linked arRaff dStrand)
    (= (time-to-arrive dStrand arRaff walking) 10)
    (= (time-to-arrive arRaff dStrand walking) 10)
    (linked dStrand arBen)
    (linked arBen dStrand)
    (= (time-to-arrive dStrand arBen walking) 11)
    (= (time-to-arrive arBen dStrand walking) 11)
    (linked dStrand arGiulio)
    (linked arGiulio dStrand)
    (= (time-to-arrive dStrand arGiulio walking) 12)
    (= (time-to-arrive arGiulio dStrand walking) 12)
    (linked arRaff arBen)
    (linked arBen arRaff)
    (= (time-to-arrive arBen arRaff walking) 1)
    (= (time-to-arrive arRaff arBen walking) 1)
    (linked arGiulio arBen)
    (linked arBen arGiulio)
    (= (time-to-arrive arBen arGiulio walking) 1)
    (= (time-to-arrive arGiulio arBen walking) 1)
    (linked arRaff arGiulio)
    (linked arGiulio arRaff)
    (= (time-to-arrive arGiulio arRaff walking) 2)
    (= (time-to-arrive arRaff arGiulio walking) 2)

    ; cluster 2
    ; cluster 2 requirements
    (= (requires arSeb egg) 2)
    (= (requires arSeb painkillers) 1)
    (= (requires arJeffrey egg) 1)
    (= (requires arJeffrey painkillers) 6)
    (= (requires arZak egg) 4)
    (= (requires arZak painkillers) 8)
    (= (requires arJay egg) 2)
    (= (requires arJay painkillers) 3)

    ; cluster 2 links
    (linked dStrand arSeb)
    (linked arSeb dStrand)
    (= (time-to-arrive dStrand arSeb walking) 7)
    (= (time-to-arrive arSeb dStrand walking) 7)
    (linked dStrand arJeffrey)
    (linked arJeffrey dStrand)
    (= (time-to-arrive dStrand arJeffrey walking) 8)
    (= (time-to-arrive arJeffrey dStrand walking) 8)
    (linked dStrand arZak)
    (linked arZak dStrand)
    (= (time-to-arrive dStrand arZak walking) 8)
    (= (time-to-arrive arZak dStrand walking) 8)
    (linked dStrand arJay)
    (linked arJay dStrand)
    (= (time-to-arrive dStrand arJay walking) 6)
    (= (time-to-arrive arJay dStrand walking) 6)
    (linked arSeb arJeffrey)
    (linked arJeffrey arSeb)
    (= (time-to-arrive arSeb arJeffrey walking) 1)
    (= (time-to-arrive arJeffrey arSeb walking) 1)
    (linked arSeb arZak)
    (linked arZak arSeb)
    (= (time-to-arrive arSeb arZak walking) 1)
    (= (time-to-arrive arZak arSeb walking) 1)
    (linked arSeb arJay)
    (linked arJay arSeb)
    (= (time-to-arrive arSeb arJay walking) 1)
    (= (time-to-arrive arJay arSeb walking) 1)
    (linked arZak arJeffrey)
    (linked arJeffrey arZak)
    (= (time-to-arrive arZak arJeffrey walking) 1)
    (= (time-to-arrive arJeffrey arZak walking) 1)
    (linked arJay arJeffrey)
    (linked arJeffrey arJay )
    (= (time-to-arrive arJay arJeffrey walking) 1)
    (= (time-to-arrive arJeffrey arJay walking) 1)
    (linked arJay arZak)
    (linked arZak arJay)
    (= (time-to-arrive arJay arZak walking) 1)
    (= (time-to-arrive arZak arJay walking) 1)

    ; cluster 3
    ; cluster 3 requirements
    (= (requires arKolling egg) 2)
    (= (requires arKolling painkillers) 1)
    (= (requires arColes egg) 1)
    (= (requires arColes painkillers) 6)
    (= (requires arKeppens egg) 4)
    (= (requires arKeppens painkillers) 8)
    (= (requires arAgi egg) 4)
    (= (requires arAgi painkillers) 8)

    ; cluster 3 links
    (linked dWaterloo arKolling)
    (linked arKolling dWaterloo)
    (= (time-to-arrive dWaterloo arKolling walking) 10)
    (= (time-to-arrive arKolling dWaterloo walking) 10)
    (linked dWaterloo arColes)
    (linked arColes dWaterloo)
    (= (time-to-arrive dWaterloo arColes walking) 11)
    (= (time-to-arrive arColes dWaterloo walking) 11)
    (linked dWaterloo arKeppens)
    (linked arKeppens dWaterloo)
    (= (time-to-arrive dWaterloo arKeppens walking) 12)
    (= (time-to-arrive arKeppens dWaterloo walking) 12)
    (linked dWaterloo arAgi)
    (linked arAgi dWaterloo)
    (= (time-to-arrive dWaterloo arAgi walking) 12)
    (= (time-to-arrive arAgi dWaterloo walking) 12)
    (linked arKolling arColes)
    (linked arColes arKolling)
    (= (time-to-arrive arKolling arColes walking) 1)
    (= (time-to-arrive arColes arKolling walking) 1)
    (linked arKolling arKeppens)
    (linked arKeppens arKolling)
    (= (time-to-arrive arKolling arKeppens walking) 1)
    (= (time-to-arrive arKeppens arKolling walking) 1)
    (linked arKolling arAgi)
    (linked arAgi arKolling)
    (= (time-to-arrive arKolling arAgi walking) 1)
    (= (time-to-arrive arAgi arKolling walking) 1)
    (linked arKeppens arColes)
    (linked arColes arKeppens)
    (= (time-to-arrive arKeppens arColes walking) 1)
    (= (time-to-arrive arColes arKeppens walking) 1)
    (linked arAgi arColes)
    (linked arColes arAgi )
    (= (time-to-arrive arAgi arColes walking) 1)
    (= (time-to-arrive arColes arAgi walking) 1)
    (linked arAgi arKeppens)
    (linked arKeppens arAgi)
    (= (time-to-arrive arAgi arKeppens walking) 1)
    (= (time-to-arrive arKeppens arAgi walking) 1)

)
```

```
(:goal (and
    ; cluster 1
    (= (requires arBen egg) 0)
    (= (requires arBen painkillers) 0)
    (= (requires arRaff egg) 0)
    (= (requires arRaff painkillers) 0)
    (= (requires arGiulio egg) 0)
    (= (requires arGiulio painkillers) 0)

    ; cluster 2
    (= (requires arSeb egg) 0)
    (= (requires arSeb painkillers) 0)
    (= (requires arJeffrey egg) 0)
    (= (requires arJeffrey painkillers) 0)
    (= (requires arZak egg) 0)
    (= (requires arZak painkillers) 0)
    (= (requires arJay egg) 0)
    (= (requires arJay painkillers) 0)

    ; cluster 3
    (= (requires arKolling egg) 0)
    (= (requires arKolling painkillers) 0)
    (= (requires arColes egg) 0)
    (= (requires arColes painkillers) 0)
    (= (requires arKeppens egg) 0)
    (= (requires arKeppens painkillers) 0)
    (= (requires arAgi egg) 0)
    (= (requires arAgi painkillers) 0)
))
)
```

Problem 5

```
(define (problem problem5) (:domain volunteerPlanner)

(:objects
    volunteer1 volunteer2 volunteer3 - volunteer
    depot1 depot2 depot3 - depot
    arPerson1 arPerson2 arPerson3 arPerson4 arPerson5 - atRiskPerson
    walking - modeOfTravel
    resource1 resource2 resource3 - resource
)

(:init
    ; resources
    (= (resource-size resource1) 0.6)
    (= (resource-size resource2) 0.2)
    (= (resource-size resource3) 1.5)

    ; depots
    (= (resources-stored resource1 depot1) 15)
    (= (resources-stored resource2 depot1) 9)
    (= (resources-stored resource3 depot1) 0)

    (= (resources-stored resource1 depot2) 2)
    (= (resources-stored resource2 depot2) 17)
    (= (resources-stored resource3 depot2) 11)

    (= (resources-stored resource1 depot3) 6)
    (= (resources-stored resource2 depot3) 5)
    (= (resources-stored resource3 depot3) 16)
    ; total for resource1 is 23
    ; total for resource2 is 31
    ; total for resource3 is 27

    ; volunteers
    (available volunteer1)
    (at volunteer3 depot1)
    (usingTransport volunteer1 walking)
    (= (capacity volunteer1 walking) 7)
    (= (activePeriod volunteer1) 40)
    (= (remainingActivePeriod volunteer1) 40)
    (= (resources-stored resource1 volunteer1) 0)
    (= (resources-stored resource2 volunteer1) 0)
    (= (resources-stored resource3 volunteer1) 0)

    (available volunteer2)
    (at volunteer2 arPerson3)
    (usingTransport volunteer2 walking)
    (= (capacity volunteer2 walking) 11)
    (= (activePeriod volunteer2) 70)
    (= (remainingActivePeriod volunteer2) 70)
    (= (resources-stored resource1 volunteer2) 0)
    (= (resources-stored resource2 volunteer2) 0)
    (= (resources-stored resource3 volunteer2) 0)

    (available volunteer3)
    (at volunteer3 arPerson3)
    (usingTransport volunteer3 walking)
    (= (capacity volunteer3 walking) 5.5)
    (= (activePeriod volunteer3) 50)
    (= (remainingActivePeriod volunteer3) 50)
    (= (resources-stored resource1 volunteer3) 0)
    (= (resources-stored resource2 volunteer3) 0)
    (= (resources-stored resource3 volunteer3) 0)

    ; links
    (linked arPerson1 arPerson2)
    (linked arPerson2 arPerson1)
    (= (time-to-arrive arPerson1 arPerson2 walking) 10)
    (= (time-to-arrive arPerson2 arPerson1 walking) 10)

    (linked arPerson2 arPerson3)
    (linked arPerson3 arPerson2)
    (= (time-to-arrive arPerson2 arPerson3 walking) 8)
    (= (time-to-arrive arPerson3 arPerson2 walking) 8)

    (linked arPerson1 depot2)
    (linked depot2 arPerson1)
    (= (time-to-arrive arPerson1 depot2 walking) 6)
    (= (time-to-arrive depot2 arPerson1 walking) 6)

    (linked depot2 arPerson2)
    (linked arPerson2 depot2)
    (= (time-to-arrive depot2 arPerson2 walking) 5)
    (= (time-to-arrive arPerson2 depot2 walking) 5)
    (linked depot1 arPerson3)
    (linked arPerson3 depot1)
    (= (time-to-arrive depot1 arPerson3 walking) 4)
    (= (time-to-arrive arPerson3 depot1 walking) 4)

    (linked depot1 arPerson4)
    (linked arPerson4 depot1)
    (= (time-to-arrive depot1 arPerson4 walking) 4)
    (= (time-to-arrive arPerson4 depot1 walking) 4)

    (linked depot2 arPerson5)
    (linked arPerson5 depot2)
    (= (time-to-arrive depot2 arPerson5 walking) 4)
    (= (time-to-arrive arPerson5 depot2 walking) 4)

    (linked depot1 depot2)
    (linked depot2 depot1)
    (= (time-to-arrive depot1 depot2 walking) 4)
    (= (time-to-arrive depot2 depot1 walking) 4)

    (linked depot3 depot2)
    (linked depot2 depot3)
    (= (time-to-arrive depot3 depot2 walking) 4)
    (= (time-to-arrive depot2 depot3 walking) 4)


    ; requirements
    (= (requires arPerson1 resource1) 6)
    (= (requires arPerson1 resource2) 0)
    (= (requires arPerson1 resource3) 7)

    (= (requires arPerson2 resource1) 3)
    (= (requires arPerson2 resource2) 9)
    (= (requires arPerson2 resource3) 0)

    (= (requires arPerson3 resource1) 12)
    (= (requires arPerson3 resource2) 5)
    (= (requires arPerson3 resource3) 8)

    (= (requires arPerson4 resource1) 0)
    (= (requires arPerson4 resource2) 11)
    (= (requires arPerson4 resource3) 4)

    (= (requires arPerson5 resource1) 3)
    (= (requires arPerson5 resource2) 6)
    (= (requires arPerson5 resource3) 8)
)

(:goal (and
    (= (requires arPerson1 resource1) 0)
    (= (requires arPerson1 resource2) 0)
    (= (requires arPerson1 resource3) 0)

    (= (requires arPerson2 resource1) 0)
    (= (requires arPerson2 resource2) 0)
    (= (requires arPerson2 resource3) 0)

    (= (requires arPerson3 resource1) 0)
    (= (requires arPerson3 resource2) 0)
    (= (requires arPerson3 resource3) 0)
    (= (requires arPerson4 resource1) 0)
    (= (requires arPerson4 resource2) 0)
    (= (requires arPerson4 resource3) 0)

    (= (requires arPerson5 resource1) 0)
    (= (requires arPerson5 resource2) 0)
    (= (requires arPerson5 resource3) 0)
))

)
```

Problem 6

```
(define (problem problem6) (:domain volunteerPlanner)
(:objects
    vWilliam, vDan - volunteer
    arRaff arBen arGiulio arJay - atRiskPerson
    dStrand - depot
    egg painkillers - resource
    walking cycling - modeOfTravel
)

(:init
    ; resources
    (= (resource-size egg) 0.2)
    (= (resource-size painkillers) 0.4)

    ; depots
    (= (resources-stored egg dStrand) 100)
    (= (resources-stored painkillers dStrand) 50)

    (at vWilliam dStrand)
    (available vWilliam)
    (usingTransport vWilliam walking)
    (= (activePeriod vWilliam) 60)
    (= (remainingActivePeriod vWilliam) 60)
    (= (capacity vWilliam walking) 10)
    (= (resources-stored egg vWilliam) 0)
    (= (resources-stored painkillers vWilliam) 0)

    (at vDan dStrand)
    (available vDan)
    (usingTransport vDan cycling)
    (= (activePeriod vDan) 60)
    (= (remainingActivePeriod vDan) 60)
    (= (capacity vDan cycling) 6)
    (= (resources-stored egg vDan) 0)
    (= (resources-stored painkillers vDan) 0)

    ; links
    ; cluster 1
    ; cluster 1 requirements
    (= (requires arRaff egg) 2)
    (= (requires arRaff painkillers) 1)
    ; cluster 1 links
    (linked arRaff dStrand)
    (linked dStrand arRaff)
    (= (time-to-arrive dStrand arRaff walking) 10)
    (= (time-to-arrive arRaff dStrand walking) 10)
    (= (time-to-arrive dStrand arRaff cycling) 3)
    (= (time-to-arrive arRaff dStrand cycling) 3)

    ; cluster 2
    ; cluster 2 requirements
    (= (requires arBen egg) 1)
    (= (requires arBen painkillers) 6)
    (= (requires arGiulio egg) 4)
    (= (requires arGiulio painkillers) 8)
    (= (requires arJay egg) 5)
    (= (requires arJay painkillers) 5)

    ; cluster 2 links
    (linked dStrand arBen)
    (linked arBen dStrand)
    (= (time-to-arrive dStrand arBen walking) 10)
    (= (time-to-arrive arBen dStrand walking) 10)
    (= (time-to-arrive dStrand arBen cycling) 3)
    (= (time-to-arrive arBen dStrand cycling) 3)
    (linked dStrand arJay)
    (linked arJay dStrand)
    (= (time-to-arrive dStrand arJay walking) 10)
    (= (time-to-arrive arJay dStrand walking) 10)
    (= (time-to-arrive dStrand arJay cycling) 3)
    (= (time-to-arrive arJay dStrand cycling) 3)
    (linked dStrand arGiulio)
    (linked arGiulio dStrand)
    (= (time-to-arrive dStrand arGiulio walking) 10)
    (= (time-to-arrive arGiulio dStrand walking) 10)
    (= (time-to-arrive dStrand arGiulio cycling) 3)
    (= (time-to-arrive arGiulio dStrand cycling) 3)
    (linked arBen arJay)
    (linked arJay arBen)
    (= (time-to-arrive arBen arJay walking) 1)
    (= (time-to-arrive arJay arBen walking) 1)
    (= (time-to-arrive arBen arJay cycling) 0.5)
    (= (time-to-arrive arJay arBen cycling) 0.5)
    (linked arBen arGiulio)
    (linked arGiulio arBen)
    (= (time-to-arrive arBen arGiulio walking) 1)
    (= (time-to-arrive arGiulio arBen walking) 1)
    (= (time-to-arrive arBen arGiulio cycling) 0.5)
    (= (time-to-arrive arGiulio arBen cycling) 0.5)
    (linked arJay arGiulio)
    (linked arGiulio arJay)
    (= (time-to-arrive arJay arGiulio walking) 1)
    (= (time-to-arrive arGiulio arJay walking) 1)
    (= (time-to-arrive arJay arGiulio cycling) 0.5)
    (= (time-to-arrive arGiulio arJay cycling) 0.5)
)

(:goal (and
    (= (requires arBen egg) 0)
    (= (requires arBen painkillers) 0)
    (= (requires arRaff egg) 0)
    (= (requires arRaff painkillers) 0)
    (= (requires arGiulio egg) 0)
    (= (requires arGiulio painkillers) 0)
    (= (requires arJay egg) 0)
    (= (requires arJay painkillers) 0)
))
)
```

Problem 7

```
(define (problem problem6) (:domain volunteerPlanner)
(:objects
    vWilliam, vDan - volunteer
    arRaff arBen arGiulio arJay - atRiskPerson
    dStrand - depot
    egg painkillers bread apple chocolate - resource
    walking - modeOfTravel
)

(:init
    ; resources
    (= (resource-size egg) 0.2)
    (= (resource-size painkillers) 0.4)
    (= (resource-size bread) 0.8)
    (= (resource-size apple) 0.1)
    (= (resource-size chocolate) 0.3)

    ; depots
    (= (resources-stored egg dStrand) 100)
    (= (resources-stored painkillers dStrand) 50)
    (= (resources-stored bread dStrand) 50)
    (= (resources-stored apple dStrand) 50)
    (= (resources-stored chocolate dStrand) 50)

    ; volunteers
    (at vWilliam dStrand)
    (available vWilliam)
    (usingTransport vWilliam walking)
    (= (activePeriod vWilliam) 60)
    (= (remainingActivePeriod vWilliam) 60)
    (= (capacity vWilliam walking) 10)
    (= (resources-stored egg vWilliam) 0)
    (= (resources-stored painkillers vWilliam) 0)
    (= (resources-stored bread vWilliam) 0)
    (= (resources-stored apple vWilliam) 0)
    (= (resources-stored chocolate vWilliam) 0)

    (at vDan dStrand)
    (available vDan)
    (usingTransport vDan walking)
    (= (activePeriod vDan) 60)
    (= (remainingActivePeriod vDan) 60)
    (= (capacity vDan walking) 6)
    (= (resources-stored egg vDan) 0)
    (= (resources-stored painkillers vDan) 0)
    (= (resources-stored bread vDan) 0)
    (= (resources-stored apple vDan) 0)
    (= (resources-stored chocolate vDan) 0)
    ; cluster 1
    ; cluster 1 requirements
    (= (requires arRaff egg) 2)
    (= (requires arRaff painkillers) 1)
    (= (requires arRaff bread) 2)
    (= (requires arRaff apple) 6)
    (= (requires arRaff chocolate) 1)
    ; cluster 1 links
    (linked arRaff dStrand)
    (linked dStrand arRaff)
    (= (time-to-arrive dStrand arRaff walking) 10)
    (= (time-to-arrive arRaff dStrand walking) 10)

    ; cluster 2
    ; cluster 2 requirements
    (= (requires arBen egg) 6)
    (= (requires arBen painkillers) 2)
    (= (requires arBen bread) 1)
    (= (requires arBen apple) 3)
    (= (requires arBen chocolate) 3)
    (= (requires arGiulio egg) 4)
    (= (requires arGiulio painkillers) 4)
    (= (requires arGiulio bread) 2)
    (= (requires arGiulio apple) 6)
    (= (requires arGiulio chocolate) 3)
    (= (requires arJay egg) 12)
    (= (requires arJay painkillers) 5)
    (= (requires arJay bread) 2)
    (= (requires arJay apple) 2)
    (= (requires arJay chocolate) 1)

    ; cluster 2 links
    (linked dStrand arBen)
    (linked arBen dStrand)
    (= (time-to-arrive dStrand arBen walking) 10)
    (= (time-to-arrive arBen dStrand walking) 10)
    (linked dStrand arJay)
    (linked arJay dStrand)
    (= (time-to-arrive dStrand arJay walking) 10)
    (= (time-to-arrive arJay dStrand walking) 10)
    (linked dStrand arGiulio)
    (linked arGiulio dStrand)
    (= (time-to-arrive dStrand arGiulio walking) 10)
    (= (time-to-arrive arGiulio dStrand walking) 10)
    (linked arBen arJay)
    (linked arJay arBen)
    (= (time-to-arrive arBen arJay walking) 1)
    (= (time-to-arrive arJay arBen walking) 1)
    (linked arBen arGiulio)
    (linked arGiulio arBen)
    (= (time-to-arrive arBen arGiulio walking) 1)
    (= (time-to-arrive arGiulio arBen walking) 1)
    (linked arJay arGiulio)
    (linked arGiulio arJay)
    (= (time-to-arrive arJay arGiulio walking) 1)
    (= (time-to-arrive arGiulio arJay walking) 1)
)

(:goal (and
    (= (requires arRaff egg) 0)
    (= (requires arRaff painkillers) 0)
    (= (requires arRaff bread) 0)
    (= (requires arRaff apple) 0)
    (= (requires arRaff chocolate) 0)
    (= (requires arBen egg) 0)
    (= (requires arBen painkillers) 0)
    (= (requires arBen bread) 0)
    (= (requires arBen apple) 0)
    (= (requires arBen chocolate) 0)
    (= (requires arGiulio egg) 0)
    (= (requires arGiulio painkillers) 0)
    (= (requires arGiulio bread) 0)
    (= (requires arGiulio apple) 0)
    (= (requires arGiulio chocolate) 0)
    (= (requires arJay egg) 0)
    (= (requires arJay painkillers) 0)
    (= (requires arJay bread) 0)
    (= (requires arJay apple) 0)
    (= (requires arJay chocolate) 0)

))
)
```

Problem 8

```
(define (problem problem_8) (:domain volunteerPlanner)
(:objects
    vWilliam - volunteer
    arRaff arBen arDan - atRiskPerson
    dStrand - depot
    egg painkillers - resource
    walking cycling - modeOfTravel
)

(:init
    ;todo: put the initial state's facts and numeric values here
    (= (resources-stored egg dStrand) 100)
    (= (resources-stored painkillers dStrand) 50)

    (at vWilliam dStrand)
    (= (activePeriod vWilliam) 100)
    (= (remainingActivePeriod vWilliam) 100)
    (available vWilliam)
    (usingTransport vWilliam walking)
    (= (capacity vWilliam walking) 10)
    (= (capacity vWilliam cycling) 6)
    (= (resources-stored egg vWilliam) 0)
    (= (resources-stored painkillers vWilliam) 0)

    (= (resource-size egg) 0.2)
    (= (resource-size painkillers) 0.4)

    (linked dStrand arRaff)
    (linked arRaff dStrand)
    (= (time-to-arrive dStrand arRaff walking) 10)
    (= (time-to-arrive arRaff dStrand walking) 10)
    (= (time-to-arrive dStrand arRaff cycling) 3)
    (= (time-to-arrive arRaff dStrand cycling) 3)
    (= (requires arRaff egg) 2)
    (= (requires arRaff painkillers) 1)

    (linked dStrand arBen)
    (linked arBen dStrand)
    (= (time-to-arrive dStrand arBen walking) 6)
    (= (time-to-arrive arBen dStrand walking) 6)
    (= (time-to-arrive dStrand arBen cycling) 2)
    (= (time-to-arrive arBen dStrand cycling) 2)
    (= (requires arBen egg) 1)
    (= (requires arBen painkillers) 6)

    (linked dStrand arDan)
    (linked arDan dStrand)
    (= (time-to-arrive dStrand arDan walking) 8)
    (= (time-to-arrive arDan dStrand walking) 8)
    (= (time-to-arrive dStrand arDan cycling) 3)
    (= (time-to-arrive arDan dStrand cycling) 3)
    (= (requires arDan egg) 2)
    (= (requires arDan painkillers) 4)
)

(:goal (and
    (= (requires arBen egg) 0)
    (= (requires arBen painkillers) 0)
    (= (requires arRaff egg) 0)
    (= (requires arRaff painkillers) 0)
    (= (requires arDan egg) 0)
    (= (requires arDan painkillers) 0)
))
)
```

Problem 9

```
(define (problem problem_9) (:domain volunteerPlanner)
(:objects
    vWilliam - volunteer
    arRaff arBen arDan arSeb - atRiskPerson
    dStrand - depot
    egg painkillers - resource
    walking cycling - modeOfTravel
)

(:init
    ;todo: put the initial state's facts and numeric values here
    (= (resources-stored egg dStrand) 100)
    (= (resources-stored painkillers dStrand) 50)

    (at vWilliam dStrand)
    (= (activePeriod vWilliam) 100)
    (= (remainingActivePeriod vWilliam) 100)
    (available vWilliam)
    (usingTransport vWilliam walking)
    (= (capacity vWilliam walking) 10)
    (= (capacity vWilliam cycling) 6)
    (= (resources-stored egg vWilliam) 0)
    (= (resources-stored painkillers vWilliam) 0)

    (= (resource-size egg) 0.2)
    (= (resource-size painkillers) 0.4)

    (linked dStrand arRaff)
    (linked arRaff dStrand)
    (= (time-to-arrive dStrand arRaff walking) 10)
    (= (time-to-arrive arRaff dStrand walking) 10)
    (= (time-to-arrive dStrand arRaff cycling) 3)
    (= (time-to-arrive arRaff dStrand cycling) 3)
    (= (requires arRaff egg) 2)
    (= (requires arRaff painkillers) 1)

    (linked dStrand arBen)
    (linked arBen dStrand)
    (= (time-to-arrive dStrand arBen walking) 6)
    (= (time-to-arrive arBen dStrand walking) 6)
    (= (time-to-arrive dStrand arBen cycling) 2)
    (= (time-to-arrive arBen dStrand cycling) 2)
    (= (requires arBen egg) 1)
    (= (requires arBen painkillers) 6)

    (linked dStrand arSeb)
    (linked arSeb dStrand)
    (= (time-to-arrive dStrand arSeb walking) 4)
    (= (time-to-arrive arSeb dStrand walking) 4)
    (= (time-to-arrive dStrand arSeb cycling) 1)
    (= (time-to-arrive arSeb dStrand cycling) 1)
    (= (requires arSeb egg) 5)
    (= (requires arSeb painkillers) 2)

    (linked dStrand arDan)
    (linked arDan dStrand)
    (= (time-to-arrive dStrand arDan walking) 8)
    (= (time-to-arrive arDan dStrand walking) 8)
    (= (time-to-arrive dStrand arDan cycling) 3)
    (= (time-to-arrive arDan dStrand cycling) 3)
    (= (requires arDan egg) 2)
    (= (requires arDan painkillers) 4)
)

(:goal (and
    (= (requires arBen egg) 0)
    (= (requires arBen painkillers) 0)
    (= (requires arRaff egg) 0)
    (= (requires arRaff painkillers) 0)
    (= (requires arDan egg) 0)
    (= (requires arDan painkillers) 0)
    (= (requires arSeb egg) 0)
    (= (requires arSeb painkillers) 0)
))
)
```

Problem 10

```
(define (problem problem1) (:domain volunteerPlanner)
(:objects
    vWilliam - volunteer
    arRaff arBen arDan arGiulio arZak - atRiskPerson
    dStrand - depot
    egg painkillers - resource
    walking cycling - modeOfTravel
)

(:init
    ; resources
    (= (resource-size egg) 0.2)
    (= (resource-size painkillers) 0.4)

    ; depots
    (= (resources-stored egg dStrand) 100)
    (= (resources-stored painkillers dStrand) 50)

    ; volunteers
    (at vWilliam dStrand)
    (available vWilliam)
    (= (activePeriod vWilliam) 100)
    (= (remainingActivePeriod vWilliam) 100)
    (usingTransport vWilliam walking)
    (= (capacity vWilliam walking) 10)
    (= (resources-stored egg vWilliam) 0)
    (= (resources-stored painkillers vWilliam) 0)

    ; links
    (linked dStrand arRaff)
    (linked arRaff dStrand)
    (= (time-to-arrive dStrand arRaff walking) 10)
    (= (time-to-arrive arRaff dStrand walking) 10)

    (linked dStrand arBen)
    (linked arBen dStrand)
    (= (time-to-arrive dStrand arBen walking) 6)
    (= (time-to-arrive arBen dStrand walking) 6)

    (linked dStrand arDan)
    (linked arDan dStrand)
    (= (time-to-arrive dStrand arDan walking) 3)
    (= (time-to-arrive arDan dStrand walking) 3)

    (linked dStrand arGiulio)
    (linked arGiulio dStrand)
    (= (time-to-arrive dStrand arGiulio walking) 7)
    (= (time-to-arrive arGiulio dStrand walking) 7)

    (linked dStrand arZak)
    (linked arZak dStrand)
    (= (time-to-arrive dStrand arZak walking) 11)
    (= (time-to-arrive arZak dStrand walking) 11)

    ; requirements
    (= (requires arBen egg) 1)
    (= (requires arBen painkillers) 6)
    (= (requires arRaff egg) 2)
    (= (requires arRaff painkillers) 1)
    (= (requires arDan egg) 3)
    (= (requires arDan painkillers) 4)
    (= (requires arGiulio egg) 3)
    (= (requires arGiulio painkillers) 2)
    (= (requires arZak egg) 3)
    (= (requires arZak painkillers) 4)
)

(:goal (and
    (= (requires arBen egg) 0)
    (= (requires arBen painkillers) 0)
    (= (requires arRaff egg) 0)
    (= (requires arRaff painkillers) 0)
    (= (requires arDan egg) 0)
    (= (requires arDan painkillers) 0)
    (= (requires arGiulio egg) 0)
    (= (requires arGiulio painkillers) 0)
    (= (requires arZak egg) 0)
    (= (requires arZak painkillers) 0)
))
)
```

Problem 11

```
(define (problem problem1) (:domain volunteerPlanner)
(:objects
    vWilliam - volunteer
    arRaff arBen arDan arGiulio arZak arJay - atRiskPerson
    dStrand - depot
    egg painkillers - resource
    walking cycling - modeOfTravel
)

(:init
    ; resources
    (= (resource-size egg) 0.2)
    (= (resource-size painkillers) 0.4)

    ; depots
    (= (resources-stored egg dStrand) 100)
    (= (resources-stored painkillers dStrand) 50)

    ; volunteers
    (at vWilliam dStrand)
    (available vWilliam)
    (= (activePeriod vWilliam) 100)
    (= (remainingActivePeriod vWilliam) 100)
    (usingTransport vWilliam walking)
    (= (capacity vWilliam walking) 10)
    (= (resources-stored egg vWilliam) 0)
    (= (resources-stored painkillers vWilliam) 0)

    ; links
    (linked dStrand arRaff)
    (linked arRaff dStrand)
    (= (time-to-arrive dStrand arRaff walking) 10)
    (= (time-to-arrive arRaff dStrand walking) 10)
    (linked dStrand arBen)
    (linked arBen dStrand)
    (= (time-to-arrive dStrand arBen walking) 6)
    (= (time-to-arrive arBen dStrand walking) 6)

    (linked dStrand arDan)
    (linked arDan dStrand)
    (= (time-to-arrive dStrand arDan walking) 3)
    (= (time-to-arrive arDan dStrand walking) 3)

    (linked dStrand arGiulio)
    (linked arGiulio dStrand)
    (= (time-to-arrive dStrand arGiulio walking) 7)
    (= (time-to-arrive arGiulio dStrand walking) 7)

    (linked dStrand arZak)
    (linked arZak dStrand)
    (= (time-to-arrive dStrand arZak walking) 11)
    (= (time-to-arrive arZak dStrand walking) 11)

    (linked dStrand arJay)
    (linked arJay dStrand)
    (= (time-to-arrive dStrand arJay walking) 12)
    (= (time-to-arrive arJay dStrand walking) 12)

    ; requirements
    (= (requires arBen egg) 1)
    (= (requires arBen painkillers) 6)
    (= (requires arRaff egg) 2)
    (= (requires arRaff painkillers) 1)
    (= (requires arDan egg) 3)
    (= (requires arDan painkillers) 4)
    (= (requires arGiulio egg) 3)
    (= (requires arGiulio painkillers) 2)
    (= (requires arZak egg) 3)
    (= (requires arZak painkillers) 4)
    (= (requires arJay egg) 2)
    (= (requires arJay painkillers) 1)
)

(:goal (and
    (= (requires arBen egg) 0)
    (= (requires arBen painkillers) 0)
    (= (requires arRaff egg) 0)
    (= (requires arRaff painkillers) 0)
    (= (requires arDan egg) 0)
    (= (requires arDan painkillers) 0)
    (= (requires arGiulio egg) 0)
    (= (requires arGiulio painkillers) 0)
    (= (requires arZak egg) 0)
    (= (requires arZak painkillers) 0)
    (= (requires arJay egg) 0)
    (= (requires arJay painkillers) 0)
))
)
```

Problem 12

```
(define (problem problem1) (:domain volunteerPlanner)
(:objects
    v1 - volunteer
    ar1 ar2 ar3 ar4 ar5 ar6 ar7 - atRiskPerson
    dStrand - depot
    egg painkillers - resource
    walking cycling - modeOfTravel
)

(:init
    ; resources
    (= (resource-size egg) 0.2)
    (= (resource-size painkillers) 0.4)

    ; depots
    (= (resources-stored egg dStrand) 100)
    (= (resources-stored painkillers dStrand) 50)

    ; volunteers
    (at v1 dStrand)
    (available v1)
    (= (activePeriod v1) 100)
    (= (remainingActivePeriod v1) 100)
    (usingTransport v1 walking)
    (= (capacity v1 walking) 10)
    (= (resources-stored egg v1) 0)
    (= (resources-stored painkillers v1) 0)

    ; links
    (linked dStrand ar1)
    (linked ar1 dStrand)
    (= (time-to-arrive dStrand ar1 walking) 10)
    (= (time-to-arrive ar1 dStrand walking) 10)

    (linked dStrand ar2)
    (linked ar2 dStrand)
    (= (time-to-arrive dStrand ar2 walking) 5)
    (= (time-to-arrive ar2 dStrand walking) 5)

    (linked dStrand ar3)
    (linked ar3 dStrand)
    (= (time-to-arrive dStrand ar3 walking) 15)
    (= (time-to-arrive ar3 dStrand walking) 15)

    (linked dStrand ar4)
    (linked ar4 dStrand)
    (= (time-to-arrive dStrand ar4 walking) 12)
    (= (time-to-arrive ar4 dStrand walking) 12)

    (linked dStrand ar5)
    (linked ar5 dStrand)
    (= (time-to-arrive dStrand ar5 walking) 16)
    (= (time-to-arrive ar5 dStrand walking) 16)

    (linked dStrand ar6)
    (linked ar6 dStrand)
    (= (time-to-arrive dStrand ar6 walking) 8)
    (= (time-to-arrive ar6 dStrand walking) 8)

    (linked dStrand ar7)
    (linked ar7 dStrand)
    (= (time-to-arrive dStrand ar7 walking) 10)
    (= (time-to-arrive ar7 dStrand walking) 10)

    ; requirements
    (= (requires ar7 egg) 2)
    (= (requires ar7 painkillers) 5)
    (= (requires ar6 egg) 3)
    (= (requires ar6 painkillers) 5)
    (= (requires ar5 egg) 4)
    (= (requires ar5 painkillers) 12)
    (= (requires ar4 egg) 1)
    (= (requires ar4 painkillers) 2)
    (= (requires ar3 egg) 2)
    (= (requires ar3 painkillers) 5)
    (= (requires ar2 egg) 4)
    (= (requires ar2 painkillers) 5)
    (= (requires ar1 egg) 2)
    (= (requires ar1 painkillers) 4)
)

(:goal (and
    (= (requires ar7 egg) 0)
    (= (requires ar7 painkillers) 0)
    (= (requires ar6 egg) 0)
    (= (requires ar6 painkillers) 0)
    (= (requires ar5 egg) 0)
    (= (requires ar5 painkillers) 0)
    (= (requires ar4 egg) 0)
    (= (requires ar4 painkillers) 0)
    (= (requires ar3 egg) 0)
    (= (requires ar3 painkillers) 0)
    (= (requires ar2 egg) 0)
    (= (requires ar2 painkillers) 0)
    (= (requires ar1 egg) 0)
    (= (requires ar1 painkillers) 0)
))
)
```

Problem 13

```
(define (problem problem1) (:domain volunteerPlanner)
(:objects
    v1 - volunteer
    ar1 ar2 ar3 ar4 ar5 ar6 ar7 ar8 - atRiskPerson
    dStrand - depot
    egg painkillers - resource
    walking cycling - modeOfTravel
)

(:init
    ; resources
    (= (resource-size egg) 0.2)
    (= (resource-size painkillers) 0.4)

    ; depots
    (= (resources-stored egg dStrand) 200)
    (= (resources-stored painkillers dStrand) 100)

    ; volunteers
    (at v1 dStrand)
    (available v1)
    (= (activePeriod v1) 100)
    (= (remainingActivePeriod v1) 100)
    (usingTransport v1 walking)
    (= (capacity v1 walking) 20)
    (= (resources-stored egg v1) 0)
    (= (resources-stored painkillers v1) 0)

    ; links
    (linked dStrand ar1)
    (linked ar1 dStrand)
    (= (time-to-arrive dStrand ar1 walking) 10)
    (= (time-to-arrive ar1 dStrand walking) 10)

    (linked dStrand ar2)
    (linked ar2 dStrand)
    (= (time-to-arrive dStrand ar2 walking) 5)
    (= (time-to-arrive ar2 dStrand walking) 5)

    (linked dStrand ar3)
    (linked ar3 dStrand)
    (= (time-to-arrive dStrand ar3 walking) 15)
    (= (time-to-arrive ar3 dStrand walking) 15)

    (linked dStrand ar4)
    (linked ar4 dStrand)
    (= (time-to-arrive dStrand ar4 walking) 12)
    (= (time-to-arrive ar4 dStrand walking) 12)

    (linked dStrand ar5)
    (linked ar5 dStrand)
    (= (time-to-arrive dStrand ar5 walking) 16)
    (= (time-to-arrive ar5 dStrand walking) 16)

    (linked dStrand ar6)
    (linked ar6 dStrand)
    (= (time-to-arrive dStrand ar6 walking) 8)
    (= (time-to-arrive ar6 dStrand walking) 8)

    (linked dStrand ar7)
    (linked ar7 dStrand)
    (= (time-to-arrive dStrand ar7 walking) 10)
    (= (time-to-arrive ar7 dStrand walking) 10)

    (linked dStrand ar8)
    (linked ar8 dStrand)
    (= (time-to-arrive dStrand ar8 walking) 7)
    (= (time-to-arrive ar8 dStrand walking) 7)

    ; requirements
    (= (requires ar8 egg) 2)
    (= (requires ar8 painkillers) 10)
    (= (requires ar7 egg) 2)
    (= (requires ar7 painkillers) 5)
    (= (requires ar6 egg) 3)
    (= (requires ar6 painkillers) 5)
    (= (requires ar5 egg) 4)
    (= (requires ar5 painkillers) 12)
    (= (requires ar4 egg) 1)
    (= (requires ar4 painkillers) 2)
    (= (requires ar3 egg) 2)
    (= (requires ar3 painkillers) 5)
    (= (requires ar2 egg) 4)
    (= (requires ar2 painkillers) 5)
    (= (requires ar1 egg) 2)
    (= (requires ar1 painkillers) 4)
)

(:goal (and
    (= (requires ar8 egg) 0)
    (= (requires ar8 painkillers) 0)
    (= (requires ar7 egg) 0)
    (= (requires ar7 painkillers) 0)
    (= (requires ar6 egg) 0)
    (= (requires ar6 painkillers) 0)
    (= (requires ar5 egg) 0)
    (= (requires ar5 painkillers) 0)
    (= (requires ar4 egg) 0)
    (= (requires ar4 painkillers) 0)
    (= (requires ar3 egg) 0)
    (= (requires ar3 painkillers) 0)
    (= (requires ar2 egg) 0)
    (= (requires ar2 painkillers) 0)
    (= (requires ar1 egg) 0)
    (= (requires ar1 painkillers) 0)
))
)
```