Efficient Shopping Report

4CCS1IAI – Introduction to Artificial Intelligence

Maria Nicolae 1606418 / K1630489

Alin Fulga 1635032 / K1630539

Henry Valeyre 1630918 / K1630490

Dimitris Papatheodoulou 1545299 / K1630513

Introduction

The last decade has obviously been a race to integrate technology into almost all daily activities. Just when you think there is nothing else to create, somebody pushes the boundaries even further. Our planning domain focuses on an elementary activity of any human being: doing shopping. The implementation offers an optimal solution for minimising the final cost of the products.

Our domain is structured in a way such that it can be easily extended to a larger scale. First, we took a very basic example of a shopping list and declared 3 markets at distinct locations. To quantify all costs on the same scale, we have added the number of kilometres to the total cost of the shopping list, considering each kilometre one pound (£1). The challenging part of this project is to find the efficient way to do your shopping, taking into consideration both distance and price.

An optional feature may be that the driver can take multiple people with him at shopping. Although they may have different shopping list the PDDL domain will work in an equivalent way. The program will find the best solution of choosing whose shopping list will be analysed first.

Relaxed Situation

To achieve an acceptable heuristic for the problem, we included the distance of the journey in the total cost. We also cogitated that the car has no fuel limit, meaning that the distance driven is not high enough to be taken into account (the vehicle never needs to be charged). However, examining a real-life situation, returning home because of the lack of boot capacity is a necessity, therefore, it is included in our implementation.

Each product has its own volume, meaning that the action of buying an item has as result an update in the boot capacity left. Since there exists the possibility of runing out of boot capacity, when the customer gets home, he or she moves the products from the vehicle in the house and we assign the boot capacity left to the actual boot capacity of the car.

There are also limitations that we took to assure that the problems are solved in an optimal time of execution. Using the OPTIC planner, we have noticed that the “-N” tag is used for giving a solution without minimizing the cost, which made us search for a better tag to fit our problem, therefore we are using the “-E” tag. Because of using the “-E” tag, the number of possibilities for the program needs to be significantly lower. On the other hand, the advantage is that the planner outputs the best possible solution at the end.

The aforementioned limitations include having a single customer for each problem file and having a single home for the customer. In practice, having more than one customer for one problem file can be redundant, if we think the program as being used by a single person, because there would not be the case of putting more than one shopping list at once. Thus, using limitations reduces the complexity of the program and produces an optimal result in an efficient time.

Implemented in our Model

- Travelling from one place to another

- A customer with a car that has a boot capacity (if the capacity is full it will have to empty it and come back

- Fuel consumed by the car from one distance to another

- Ability of a customer to buy a product from a shop at a certain price

- The customer can go Home

Objects

Location – There will be in our problem different location that can be shops , home or normal location

Vehicle - A customer will have a vehicle with a boot capacity to drive to places

Customer – A customer will be the one shopping products . A customer has a name

Product – Each products have a name , and are available at different quantities in a different shop and they will have different prices.

Predicates

-isAvailable ?x - Location ?p – Product : says if a product is available at a location

-hasRoute ?x - Location ?y – Location : says if there is a route/ way from one location to another

-isMarket ?x – Location : says if a location is a market or not

-isHome ?x – Location : says if the location is the home

-isAtHome ?c – Customer : checks if the customer is at home

-isAtLocation ?x - Location ?c – Customer : says at which location the customer is

-isInVehicle ?c - Customer ?v – Vehicle : says which vehicle the customer is in .

Functions

-quantityProd ?x - Location ?p – Product : This will indicate the quantity of a product at a shop

-bootCapacity ?v – Vehicle : This will indicate in litres the boot volume and capacity of a car

-bootVolumeLeft ?v – Vehicle : This is the volume left , volume that will decrease by the product’s volume

-volumeProduct ?p - Product ?x – Location : This will indicate the volume in Litres of a product

-cost : We relaxed the cost as distance+product's price

-money ?c – Customer : this will indicate the amount of money a customer has

-costLocation ?x - Location ?y – Location : this indicates how much it costs to move from one location to another

-costProduct ?p - Product ?x – Location : This is the price of a product in a shop , it will depend of the shop

-purchaseProd ?p - Product ?c – Customer : this indicates how much of a product a customer needs to purchase and it will thus help us to determin the goal of the program as it is to have 0 products on the list.

Actions

There is 3 different actions . Each action is a durative action as it takes place along the program.

moveToLocation - It defines the action to move a vehicle from a point A to point B. It has a duration which is the distance between those two locations. Every time the vehicle moves the cost increases by the distance from the next location.

releaseProducts - It defines the action to move and release all purchased products at home. It has a duration which is identified by the distance of the two locations. Checks whether the vehicle is at home in the end and releases the products.

buyProduct – It defines the action to buy a product from the market. It has a duration which is the cost of the product from that market. Every time a product is purchased the boot's volume decreases by the volume of each product.

Extensions

Continuously trying to make the domain as close to real life as possible can lead to extensions of the current program. One of them would be the capability of the customer to enter/exit a vehicle, that we considered unnecessary for this domain, at the current level of development. Another extension would imply the customer having more than one shopping list, including the order that those shopping lists are prioritised.

Conclusions

Being a very common activity, shopping became a habit. Thus, to be as quick as possible, everyone omits taking into account the amount of money they spent and go to the closest market. The consequences of their action have a significant impact on their financial situation in time. Our PDDL domain finds the right balance between time and cost. For instance, in our problem files the customers go to multiple shops to optimize the expenditures.

Appendix

Visual representation of problems

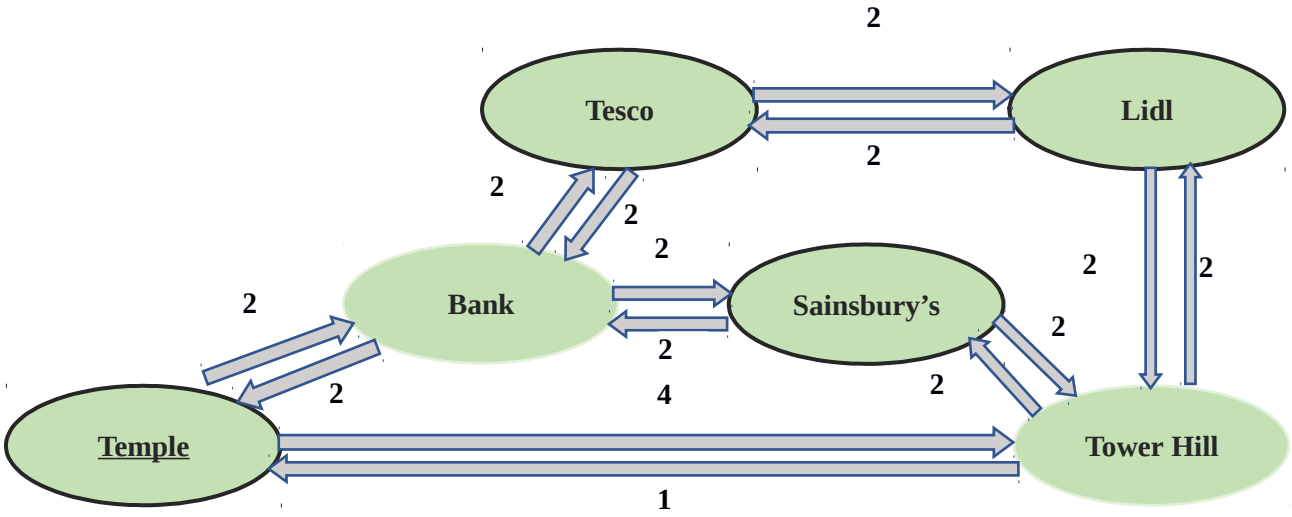
Shop Stock (problems are differentiated in the table by the background colour)

ID	Product	Shop	Cost	Volume (L)	Quantity
1.	Egg	Lidl	0.790	3.53	4
		Tesco	1.750	3.57	8
2.	Flour	Lidl	0.450	1.430	2
		Tesco	0.700	1.450	3
		Sainsbury’s	0.600	1.500	1
3.	Chicken	Tesco	5.800	3.760	10
		Iceland	5.000	3.876	12
		Lidl	5.590	3.950	3
4.	Tomatoes	Tesco	0.900	0.97	10
		Iceland	1.100	1	15
		Lidl	0.490	0.90	20
5.	Choco Muffin	Tesco	1.000	0.35	14
		Iceland	1.000	0.40	14
		Lidl	0.490	0.42	14
6.	Frozen Fruits	Tesco	3.000	0.64	6
		Lidl	1.990	0.68	8

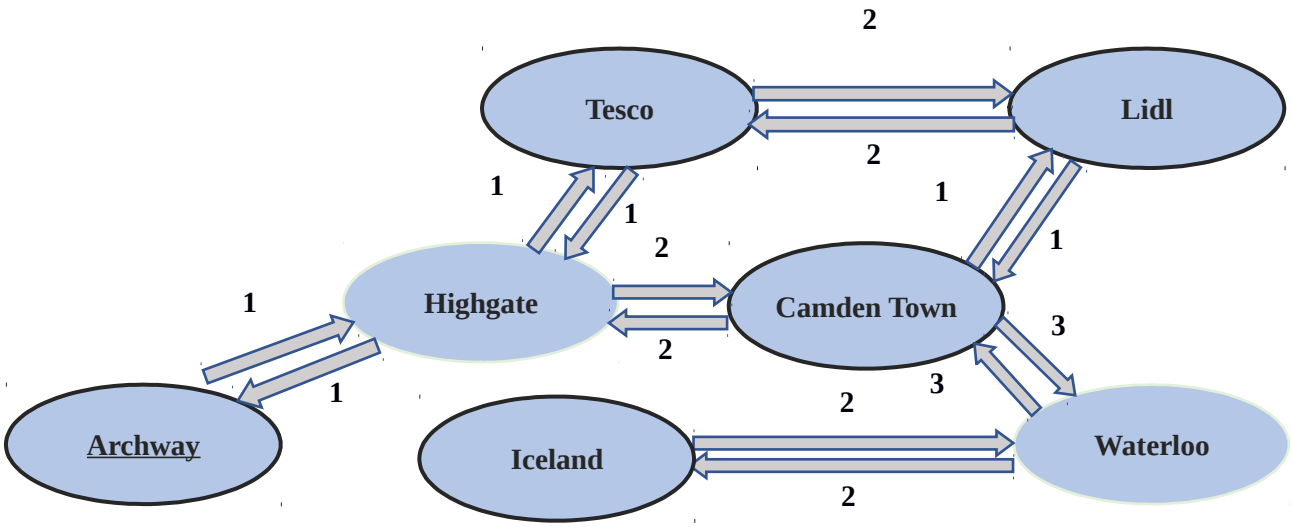
Shopping list (problems are differentiated in the table by the background colour)

ID	Customer	Product	Quantity
1.	Mario	Egg	12
2.	Mario	Flour	4
3.	Maria	Chicken	5
4.	Maria	Tomatoes	5
5.	Maria	Choco Muffin	10
6.	Maria	Frozen Fruits	1

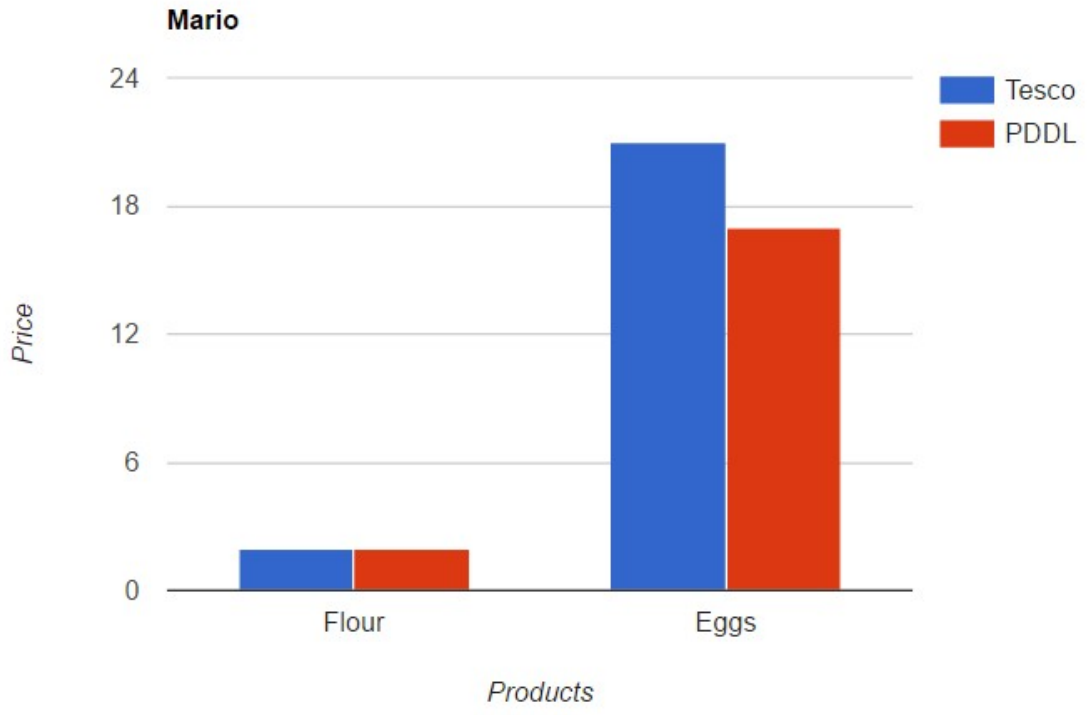
Road Map (Problem 1) – includes cost to go to a certain location, markets and home location are highlighted



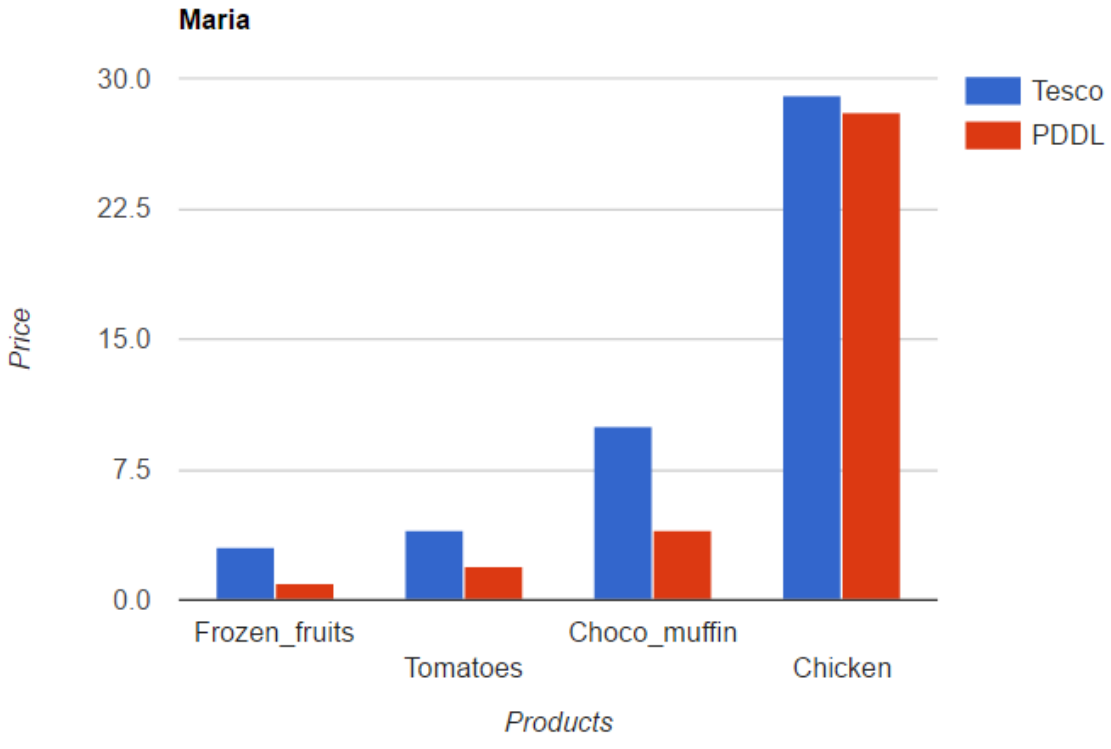
Road Map (Problem 2) – includes cost to go to a certain location, markets and home location are highlighted



Bar Graph (problem 1) – It shows the difference between the cost for each product found in average in the shops by our PDDL program and the cost for each product in the Tesco shop.



Bar Graph (problem 2) – It shows the difference between the cost for each product found in average in the shops by our PDDL program and the cost for each product in the Tesco shop.



Plan for first Problem File

```
; Plan found with metric 28.460
; Theoretical reachable cost 28.460
; States evaluated so far: 33070
; States pruned based on pre-heuristic cost lower bound: 17037
; Time 34.85
0.000: (movetolocation temple bank mario bmwx6)  [2.000]
2.001: (movetolocation bank tesco mario bmwx6)  [2.000]
4.001: (buyproduct tesco flour mario bmwx6)  [0.700]
4.003: (buyproduct tesco flour mario bmwx6)  [0.700]
4.004: (buyproduct tesco eggs mario bmwx6)  [1.750]
4.005: (buyproduct tesco eggs mario bmwx6)  [1.750]
4.006: (buyproduct tesco eggs mario bmwx6)  [1.750]
4.007: (buyproduct tesco eggs mario bmwx6)  [1.750]
4.008: (buyproduct tesco eggs mario bmwx6)  [1.750]
4.009: (buyproduct tesco eggs mario bmwx6)  [1.750]
4.010: (buyproduct tesco eggs mario bmwx6)  [1.750]
4.011: (buyproduct tesco eggs mario bmwx6)  [1.750]
5.761: (movetolocation tesco lidl mario bmwx6)  [2.000]
7.761: (buyproduct lidl flour mario bmwx6)  [0.450]
7.762: (buyproduct lidl flour mario bmwx6)  [0.450]
7.763: (buyproduct lidl eggs mario bmwx6)  [0.790]
7.764: (buyproduct lidl eggs mario bmwx6)  [0.790]
7.765: (buyproduct lidl eggs mario bmwx6)  [0.790]
7.766: (buyproduct lidl eggs mario bmwx6)  [0.790]
8.556: (movetolocation lidl towerhill mario bmwx6)  [2.000]
10.557: (releaseproducts towerhill temple mario bmwx6)  [1.000]
```

In the first problem file, the customer Mario drives an BMX X6 and has its home set at Temple. To arrive at the first shop, it moves to Bank, and then it can enter Tesco, and buy flour and eggs. Since the amount of eggs in Tesco is lower than the quantity needed by the customer, he moves to Lidl and buys the rest of flour and eggs. Then, it moves to Tower Hill and then to Temple, where he releases the products and the plan finishes.

Plan for second Problem File

```
; Plan found with metric 47.710
; Theoretical reachable cost 47.710
; States evaluated so far: 538
; States pruned based on pre-heuristic cost lower bound: 43
; Time 1.21
0.000: (movetolocation archway highgate maria audia3s)  [1.000]
1.001: (movetolocation highgate camdentown maria audia3s)  [2.000]
3.002: (movetolocation camdentown lidl maria audia3s)  [1.000]
4.004: (buyproduct lidl frozenfruits maria audia3s)  [1.989]
4.006: (buyproduct lidl chocomuffin maria audia3s)  [0.489]
4.008: (buyproduct lidl chocomuffin maria audia3s)  [0.489]
4.010: (buyproduct lidl chocomuffin maria audia3s)  [0.489]
4.012: (buyproduct lidl chocomuffin maria audia3s)  [0.489]
4.014: (buyproduct lidl chocomuffin maria audia3s)  [0.489]
4.016: (buyproduct lidl chocomuffin maria audia3s)  [0.489]
4.018: (buyproduct lidl chocomuffin maria audia3s)  [0.489]
4.020: (buyproduct lidl chocomuffin maria audia3s)  [0.489]
4.022: (buyproduct lidl chocomuffin maria audia3s)  [0.489]
4.024: (buyproduct lidl chocomuffin maria audia3s)  [0.489]
4.026: (buyproduct lidl tomatoes maria audia3s)  [0.489]
4.028: (buyproduct lidl tomatoes maria audia3s)  [0.489]
4.030: (buyproduct lidl tomatoes maria audia3s)  [0.489]
4.032: (buyproduct lidl tomatoes maria audia3s)  [0.489]
4.034: (buyproduct lidl tomatoes maria audia3s)  [0.489]
4.036: (buyproduct lidl chicken maria audia3s)  [5.589]
4.037: (buyproduct lidl chicken maria audia3s)  [5.589]
4.039: (buyproduct lidl chicken maria audia3s)  [5.589]
9.628: (movetolocation lidl camdentown maria audia3s)  [1.000]
10.629: (movetolocation camdentown highgate maria audia3s)  [2.000]
12.630: (movetolocation highgate tesco maria audia3s)  [1.000]
13.630: (buyproduct tesco chicken maria audia3s)  [5.799]
13.631: (buyproduct tesco chicken maria audia3s)  [5.799]
19.430: (movetolocation tesco highgate maria audia3s)  [1.000]
20.431: (releaseproducts highgate archway maria audia3s)  [1.000]
```

In the second problem file, the customer is Maria, she drives an Audi A3S and has her home at Archway. She moves to Highgate, and then to Camden Town, in order to get to Lidl. From Lidl, she buys choco muffins, tomatoes, chicken and frozen fruits. Then, she goes back to Camden Town, and then to Highgate, but stops at Tesco. Even if Lidl is the best market for the products that she needs, it does not have enough products, so she has to buy the remaining chicken from Tesco, and then go back to Highgate and stop at Archway, where she releases the products and the plan stops.

Domain File (market_system.pddl)

```
(define (domain market_system)
(:requirements
        :typing :durative-actions :numeric-fluents :equality)
;----Declaring types of variables-----
(:types
        Location
        Vehicle
     Customer
        Product
)
;----Declaring predicates using different types of variables----
(:predicates
        (isAvailable ?x - Location ?p - Product)
        (hasRoute ?x - Location ?y - Location)
        (isMarket ?x - Location)
        (isHome ?x - Location)
        (isAtHome ?c - Customer)
        (isAtLocation ?x - Location ?c - Customer)
        (isInVehicle ?c - Customer ?v - Vehicle)
)
;----Defining functions that can be applied by using specific types of variables----
(:functions
        (quantityProd ?x - Location ?p - Product)
        (bootCapacity ?v - Vehicle)
        (bootVolumeLeft ?v - Vehicle)
        (volumeProduct ?p - Product ?x - Location)
        (cost) - number;---We relaxed the cost as distance+product's price
        (money ?c - Customer)
        (costLocation ?x - Location ?y - Location)
        (costProduct ?p - Product ?x - Location)
        (purchaseProd ?p - Product ?c - Customer)
)
```

; Will be continued the next page.

```
;----Defining the action to move a vehicle from a point A to point B. It has a duration which is the distance between those two locations. Every time the vehicle moves the cost increases by the distance from the next location-----
(:durative-action moveToLocation
    :parameters (?x - Location ?y - Location ?c - Customer ?v - Vehicle)
    :duration (= ?duration (costLocation ?x ?y))
    :condition (and (at start (isAtLocation ?x ?c))
                    (at start (isInVehicle ?c ?v))
                                        (over all (hasRoute ?x ?y)));---For now we don't consider fuel cost
    :effect (and (at start (not (isAtLocation ?x ?c)));---Maybe can be substituted by 'over all'
                        (at end (not (isAtLocation ?x ?c)))
                        (at end (not (isAtHome ?c)))
                        (at end (isAtLocation ?y ?c));----We don't care for fuel for the moment (we decrease if we had it)
                        (at start (increase (cost)(costLocation ?x ?y))))
)
;-----Defining the action to move and release all purchased products at home. It has a duration which is identified by the distance of the two locations. Checks whether the vehicle is at home in the end and releases the products-----
(:durative-action releaseProducts
    :parameters (?x - Location ?y - Location ?c - Customer ?v - Vehicle)
    :duration (= ?duration (costLocation ?x ?y))
    :condition (and (at start (isAtLocation ?x ?c))
                    (at start (isInVehicle ?c ?v))
                                        (over all (hasRoute ?x ?y))
                                        (at start (isHome ?y)));---For now we don't consider fuel cost
    :effect (and (at start (not (isAtLocation ?x ?c)));---Maybe can be substituted by 'over all'
                        (at end (not (isAtLocation ?x ?c)))
                        (at end (isAtHome ?c))
                        (at end (isAtLocation ?y ?c));----We don't care for fuel for the moment (we decrease if we had it)
                        (at start (increase (cost)(costLocation ?x ?y)))
                        (at start (assign (bootVolumeLeft ?v) (bootCapacity ?v))))
)
```

; Will be continued the next page.

```
;-----Defining the action to buy a product from the market. It has a duration which is the cost of the product from that market. Every time a product is purchased the boot's volume decreases by the volume of each product-----

(:durative-action buyProduct


        :parameters (?x - Location ?p - Product ?c - Customer ?v - Vehicle)

        :duration (= ?duration (costProduct ?p ?x))


        :condition (and (over all (isMarket ?x))
                        (over all (isInVehicle ?c ?v))
                                        (over all (isAtLocation ?x ?c))
                                        (at start (>(quantityProd ?x ?p)0))
                                        (at start (>(bootVolumeLeft ?v)
(volumeProduct ?p ?x)))
                                        (at start (>(money ?c) (costProduct ?p ?x)))
                                )


        :effect(and (at start (decrease (quantityProd ?x ?p) 1))
                    (at start (decrease (purchaseProd ?p ?c) 1))
                    (at start (decrease (money ?c) (costProduct ?p ?x)))
                    (at start (increase (cost) (costProduct ?p ?x)))
                         (at start (decrease (bootVolumeLeft ?v) (volumeProduct ?p ?
x)))


         )


)
)
```

First Problem File (buy_product.pddl)

```
(define (problem buy_product)

(:domain market_system)

(:objects

        BMWX6 - Vehicle
        Mario - Customer
        Temple Bank Sainsburys Towerhill Lidl Tesco - Location
        Flour Eggs - Product )

(:init

   (= (bootCapacity BMWX6) 580)
        (= (bootVolumeLeft BMWX6) 580)
        (isMarket Sainsburys)
        (isMarket Lidl)
        (isMarket Tesco)
        (isHome Temple)
        (= (cost) 0)
        (isAtLocation Temple Mario)
        (isInVehicle Mario BMWX6)

        (hasRoute Temple Bank)
        (= (costLocation Temple Bank) 2)
        (hasRoute Bank Temple)
        (= (costLocation Bank Temple) 2)

        (hasRoute Bank Sainsburys)
        (= (costLocation Bank Sainsburys) 2)
        (hasRoute Sainsburys Bank)
        (= (costLocation Sainsburys Bank) 2)
```

; Will be continued the next page.

```
        (hasRoute Sainsburys Towerhill)
        (= (costLocation Sainsburys Towerhill) 2)
        (hasRoute Towerhill Sainsburys)
        (= (costLocation Towerhill Sainsburys) 2)

        (hasRoute Towerhill Lidl)
        (= (costLocation Towerhill Lidl) 2)
        (hasRoute Lidl Towerhill)
        (= (costLocation Lidl Towerhill) 2)

        (hasRoute Temple Towerhill)
        (= (costLocation Temple Towerhill) 4)
        (hasRoute Towerhill Temple)
        (= (costLocation Towerhill Temple) 1)

        (hasRoute Lidl Tesco)
        (= (costLocation Lidl Tesco) 2)
        (hasRoute Tesco Lidl)
        (= (costLocation Tesco Lidl) 2)

        (hasRoute Bank Tesco)
        (= (costLocation Bank Tesco) 2)
        (hasRoute Tesco Bank)
        (= (costLocation Tesco Bank) 2)

        (= (costProduct Flour Lidl) 0.450)
        (= (volumeProduct Flour Lidl) 1.430)
        (= (quantityProd Lidl Flour) 2)

        (= (costProduct Flour Tesco) 0.700)
        (= (volumeProduct Flour Tesco) 1.450)
        (= (quantityProd Tesco Flour) 3)

        (= (costProduct Flour Sainsburys) 0.600)
        (= (volumeProduct Flour Sainsburys) 1.500)
        (= (quantityProd Sainsburys Flour) 1)
```

; Will be continued the next page.

```
        (= (costProduct Eggs Lidl) 0.790)
        (= (volumeProduct Eggs Lidl) 3.53)
        (= (quantityProd Lidl Eggs) 4)

        (= (costProduct Eggs Tesco) 1.750)
        (= (volumeProduct Eggs Tesco) 3.57)
        (= (quantityProd Tesco Eggs) 8)

        (= (purchaseProd Eggs Mario) 12)
        (= (purchaseProd Flour Mario) 4)

        (= (money Mario) 100)
)
(:goal

        (and (= (purchaseProd Eggs Mario) 0)
            (= (purchaseProd Flour Mario) 0)
            (isAtHome Mario))
)
(:metric minimize (cost))

)
```

; Will be continued the next page.

Second Problem File (buy_product1.pddl)

```
(define (problem buy_product1)

(:domain market_system)

(:objects

        AudiA3S - Vehicle
        Maria - Customer
        Archway Lidl CamdenTown Tesco Highgate Iceland Waterloo  - Location
        Chicken Breadsticks Tomatoes FrozenFruits ChocoMuffin - Product )

(:init

   (= (bootCapacity AudiA3S) 380)
        (= (bootVolumeLeft AudiA3S) 380)
        (isMarket Iceland)
        (isMarket Lidl)
        (isMarket Tesco)
        (isHome Archway)
        (= (cost) 0)
        (isAtLocation Archway Maria)
        (isInVehicle Maria AudiA3S)

        (hasRoute Archway Highgate)
        (= (costLocation Archway Highgate) 1)
        (hasRoute Highgate Archway)
        (= (costLocation Highgate Archway) 1)

        (hasRoute Highgate Tesco)
        (= (costLocation Highgate Tesco) 1)
        (hasRoute Tesco Highgate)
        (= (costLocation Tesco Highgate) 1)
```

; Will be continued the next page.

```
        (hasRoute Highgate CamdenTown)
        (= (costLocation Highgate CamdenTown) 2)
        (hasRoute CamdenTown Highgate)
        (= (costLocation CamdenTown Highgate) 2)
        (hasRoute CamdenTown Lidl)
        (= (costLocation CamdenTown Lidl) 1)
        (hasRoute Lidl CamdenTown)
        (= (costLocation Lidl CamdenTown) 1)
        (hasRoute CamdenTown Waterloo)
        (= (costLocation CamdenTown Waterloo) 3)
        (hasRoute Waterloo CamdenTown)
        (= (costLocation Waterloo CamdenTown) 3)

        (hasRoute Waterloo Iceland)
        (= (costLocation Waterloo Iceland) 2)
        (hasRoute Iceland Waterloo)
        (= (costLocation Iceland Waterloo) 2)

        (= (costProduct Chicken Tesco) 5.800)
        (= (volumeProduct Chicken Tesco) 3.760)
        (= (quantityProd Tesco Chicken) 10)

        (= (costProduct Chicken Iceland) 5.000)
        (= (volumeProduct Chicken Iceland) 3.876)
        (= (quantityProd Iceland Chicken) 12)

        (= (costProduct Chicken Lidl) 5.590)
        (= (volumeProduct Chicken Lidl) 3.950)
        (= (quantityProd Lidl Chicken) 3)

        (= (costProduct FrozenFruits Tesco) 3.000)
        (= (volumeProduct FrozenFruits Tesco) 0.64)
        (= (quantityProd Tesco FrozenFruits) 6)

        (= (costProduct FrozenFruits Lidl) 1.990)
        (= (volumeProduct FrozenFruits Lidl) 0.68)
        (= (quantityProd Lidl FrozenFruits) 8)
```

; Will be continued the next page.

```
        (= (costProduct ChocoMuffin Tesco) 1.000)
        (= (volumeProduct ChocoMuffin Tesco) 0.35)
        (= (quantityProd Tesco ChocoMuffin) 14)
        (= (costProduct ChocoMuffin Iceland) 1.000)
        (= (volumeProduct ChocoMuffin Iceland) 0.40)
        (= (quantityProd Iceland ChocoMuffin) 14)
        (= (costProduct ChocoMuffin Lidl) 0.490)
        (= (volumeProduct ChocoMuffin Lidl) 0.42)
        (= (quantityProd Lidl ChocoMuffin) 14)
        (= (costProduct Tomatoes Tesco) 0.900)
        (= (volumeProduct Tomatoes Tesco) 0.97)
        (= (quantityProd Tesco Tomatoes) 10)

        (= (costProduct Tomatoes Iceland) 1.100)
        (= (volumeProduct Tomatoes Iceland) 1)
        (= (quantityProd Iceland Tomatoes) 15)

        (= (costProduct  Tomatoes Lidl) 0.490)
        (= (volumeProduct Tomatoes Lidl) 0.90)
        (= (quantityProd Lidl Tomatoes) 20)
        (= (purchaseProd Chicken Maria) 5)
        (= (purchaseProd Tomatoes Maria) 5)
        (= (purchaseProd ChocoMuffin Maria) 10)
        (= (purchaseProd FrozenFruits Maria) 1)
        (= (money Maria) 100)
)
(:goal
        (and (= (purchaseProd Chicken Maria) 0)
                (= (purchaseProd Tomatoes Maria) 0)
                (= (purchaseProd ChocoMuffin Maria) 0)
                (= (purchaseProd FrozenFruits Maria) 0)
             (isAtHome Maria))
)
(:metric minimize (cost))
)
```