Attentec

Ruttplanering för särskild kollektivtrafik

Utvärdering av möjliga lösningar

2020-08-17

Kammakargatan 22
111 40 Stockholm

Teknikringen 4B
583 30 Linköping

Bogstadveien 54
0366 Oslo

Innehållsförteckning

1. Inledning

Det här dokumentet innehåller och sammanfattar lärdomar och erfarenheter från förstudien ‘Ruttplanering’ som var ett samarbete mellan Östgötatrafiken och Attentec. Tanken med rapporten är att dokumentera och förklara de väsentligaste delarna av förstudien för att detta skall kunna ligga till grund för fortsatt arbete kring frågor som rör ruttplanering.

1.1. Bakgrund

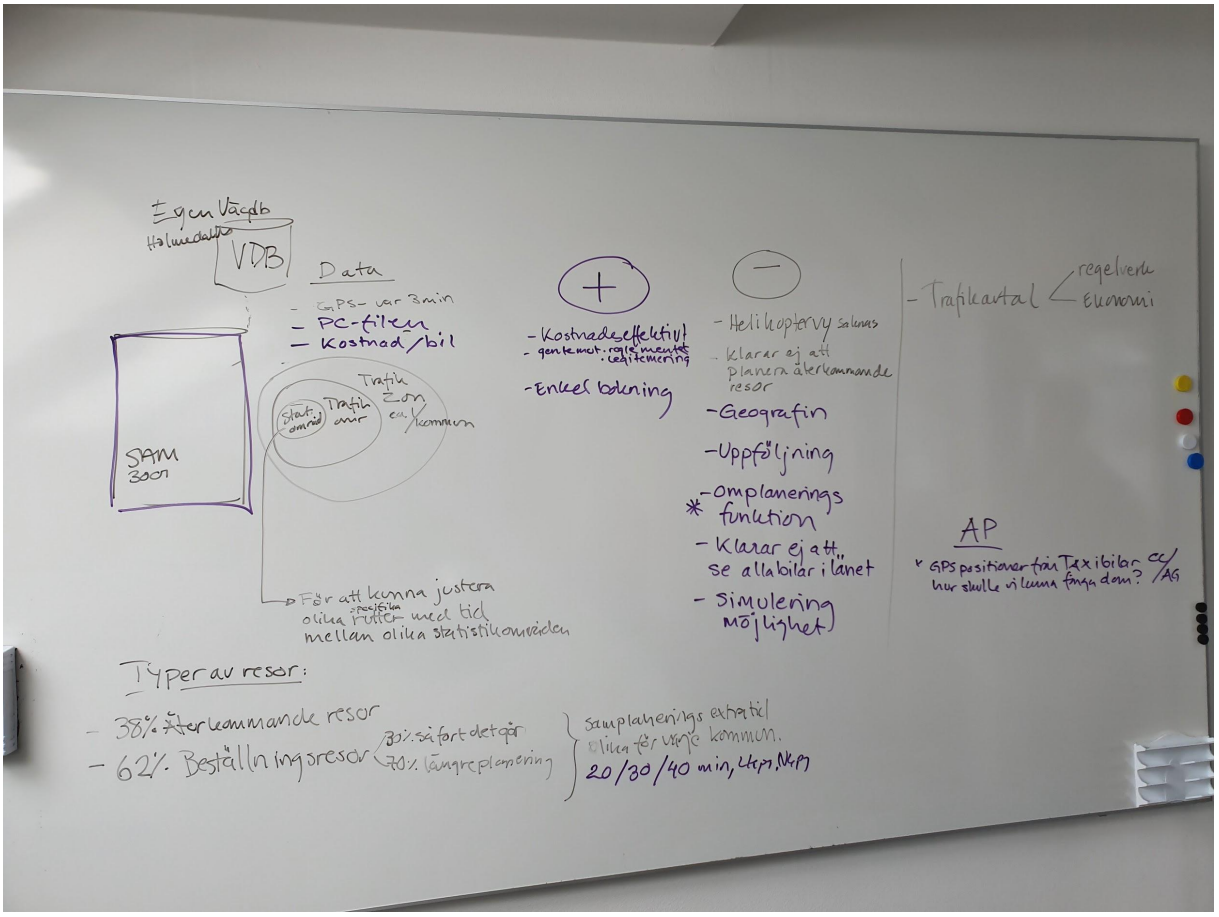
Östgötatrafiken ansvarar för beställningstrafik av olika typer i hela Östergötland. Några typer av beställningstrafik är färdtjänst, sjukresor och skolskjuts, där ungefär 38% är så kallade återkommande resor och 62% beställningsresor. Ungefär 5500 - 5600 resor genomförs dagligen vilket ger en siffra på cirka 1.4 miljoner genomförda resor på år. Östgötatrafiken har för tillfället upphandlat ett hundratal fordon med i huvudsak tre olika avtal som används för att genomföra resorna.

Det finns en hel del regler att ta hänsyn till för att kunna genomföra en beställningsresa. Dels måste resenären vara berättigad resan, vilket kontrolleras med en så kallad *legitimering*. Vid legitimeringen kan också resenären ha behov som påverkar vilken typ av fordon resenären får åka med. Det kan exempelvis handla om behov av att ha med rullstol, ledsagare eller att extra hänsyn bör tas genom att resenären måste åka ensam i fordonet.

När legitimeringen är klar och förutsättningarna för resan är fastställda sker en bokning av resan. Dessa bokade resor planeras sedan ut på tillgängliga fordon på ett sådant sätt att eventuella behov hos resenären uppfylls, samt att regelverket för när en resenär får bli hämtad/lämnad följs. Målet med planen är att hålla nere den totala kostnaden och undvika ‘onödiga’ resor, vilket kan ske med bl.a. samplanering. I de fall där flera resenärer kan åka med ett fordon slås kostnaden ut på fler resenärer vilket minskar just kostnaden per resenär. Självklart finns krav på hur lång en omväg får vara för en resenär (30 minuter) som också måste tas hänsyn till i dessa fall.

Det stora antalet beställningsresor sätter krav på att planeringen måste kunna vara flexibel för att kunna ta hänsyn till eventuella ändringar och/eller nya bokningar. I dagsläget använder Östgötatrafiken ett och samma system för att hantera bokningar av resor inklusive legitimering samt planering av rutterna. Dagens system kräver del manuell handpåläggning av personer med domänkunskap för att skapa fungerande och bra ruttplaner.

Figuren nedan visar några av de tankar och idér som diskuterades under den inledande workshopen som användes för att hitta intressanta infallsvinklar på problem kopplade till ruttplanering i den skala som Östgötatrafiken jobbar med, vilket blev projektets utgångspunkt.



Tankar och idér från den inledande workshopen.

Under workshopen identifierades några områden som i dagsläget upplevs fungera bra respektive mindre bra. Det blev också tydligt att det mest intressanta spåret att titta närmare på var hur *AI-planering*[1] (som också benämns *automatisk planering*) skulle kunna användas för att förbättra delar som i dagsläget fungerar mindre bra.

Delar som fungerar bra:

- Kostnadseffektiva rutter
    - Tar hänsyn till reglemente och legitimering
- Enkel bokning

Delar som fungerar mindre bra:

- Svårt att överblicka systemet
- Klarar ej att planera återkommande resor
- Geografi
    - Estimering av körsträcka och tid
- Uppföljning/utvärdering
- Förmåga att planera om

[1] https://planning.wiki/

- Inkludera alla tillgängliga fordon i planerna, oavsett län
- Möjlighet att simulera/testa olika situationer

1.2. Syfte

Förstudiens syfte är att undersöka och testa hur nya tekniker inom framför allt AI kan användas och tillämpas för att förbättra och underlätta arbetet med planeringen av rutter. Genom att applicera moderna tekniker på problemet är också tanken att erfarenheter och slutsatser kan bidra till hur framtidens ruttplaneringssystem skulle kunna se ut.

1.3. Mål

Utvärdera och testa fördelar och nackdelar med att applicera automatisk planering för Östgötatrafikens behov av ruttplanering.

2. Automatisk planering

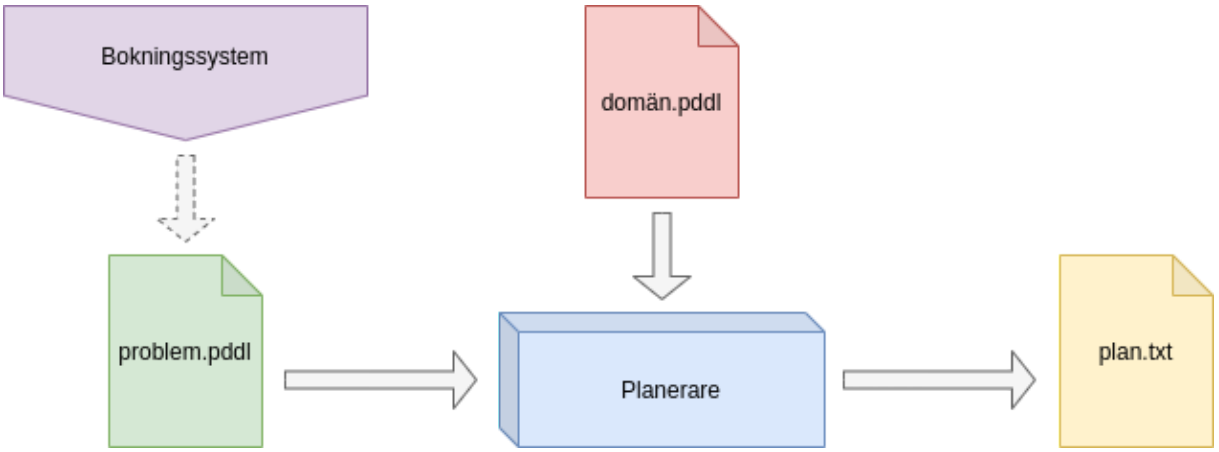
Automatisk planering är ett pågående forskningsområde inom *artificiell intelligens* (AI), med kopplingar mot schemaläggning och optimering[2]. I grund och botten handlar automatisk planering om att hitta en sekvens av handlingar (plan) som uppfyller ett *måltillstånd*, givet ett *nuvarande tillstånd*. Tanken är att beskriva hur världen ser ut, hur vi vill att den skall se ut samt vad som går att göra i världen, och låta en planerare söka reda på en sekvens av lämpliga handlingar med hjälp av olika, smarta söktekniker. Det ger oss en rad positiva fördelar i fallet med automatisk ruttplanering:

- Inget beroende av en specifik geografi
    - Den uppskattade körsträckan och körtiden mellan två platser görs av tredje part
- Enkelt att planera om
- Tar hänsyn till alla fordon och resenärer och skapar en övergripande plan
- Möjligheten till simuleringar
    - Genom att skapa fabricerade situationer och sedan utvärdera vad planeraren kommer fram till

Det finns såklart väldigt många olika planeringsproblem och för att, på ett generellt sätt, kunna beskriva ett planeringsproblem används ett språk som kallas PDDL (Planning Domain Definition Language)[3]. Under årens lopp har språket utvecklas för att kunna beskriva lite mer komplexa problem[4]. På så sätt blir det också enkelt att använda olika typer av *planerare*. Planeraren kan använda olika taktiker för att försöka lösa det problem som beskrivs i PDDL.

[2] https://nms.kcl.ac.uk/planning/index.html
[3] https://planning.wiki/_citedpapers/pddl1998.pdf
[4] https://planning.wiki/_citedpapers/pddl222004.pdf

Det finns en uppsjö av olika planerare[5], med lite olika egenskaper. I huvudsak kan man säga att en planerare ansvarar för att skapa en plan, givet en *domän* och ett *problem*. Se figuren nedan.



Överblick av grunderna i automatisk planering.

Domänfilen (domain.pddl i exemplet nedan) specificerar i stora drag spelreglerna för vad som går att göra. Mer konkret innehåller domänen:

- Typer av objekt och dess relationer
  - Fordon av typen taxi, SPEC
- Möjliga logiska påståenden
  - Fordonet *F* är i garaget, passageraren är på adress X
- Möjliga handlingar och dess konsekvenser
  - Kör fordonet från punkt A till punkt B, hämta resenär X, lämna resenär X
  -

Om exempelvis handlingen 'kör fordon *F* från punkt *A* till punkt *B'* utförs, får det konsekvensen att fordon *F* inte längre är på punkt *A*, utan på punkt *B* efter det att handlingen har utförts. Notera att domänen inte specificerar konkreta platser eller fordon, utan bara att det exempelvis går att köra ett fordon mellan två platser. På det sättet skapar vi en regler att förhålla oss till och som sedan planeraren måste ta hänsyn till i sökningen efter en plan.

Problemfilen (problem.pddl i exemplet ovan) definierar vilka objekt som finns, det nuvarande tillståndet samt måltillståndet. Det innebär att vi bl.a. konkretiserar vilka platser som finns (där resenärerna skall hämtas och lämnas), hur långt det är mellan platserna, uppskattad restid mellan platserna, vilka resenärer som finns, vilka behov dessa resenärer har (ledsagare,

[5] https://planning.wiki/ref/planners

eventuella hjälpmedel o.s.v.), när resenären kan bli hämtad och lämnad och vart vi vill att alla fordon och resenärer skall befinna sig när vi är klara. Typiskt sett har den här informationen mycket att göra med vilka bokningar som ligger inne vilket gör att det ofta är praktiskt att generera problemfilen baserat på bokningsläget. Det är sedan upp till planeraren att hitta en serie av lämpliga handlingar som resulterar i att vårt måltillstånd uppfylls. Notera att det teoretiskt sett är väldigt enkelt att lägga till och ta bort resor ur problemfilen vid behov för att på så sätt försöka hitta en plan som tar hänsyn till nya förutsättningar.

Automatisk planering är uppdelat i lite olika områden, där så kallad klassisk planering beskriver det allra enklaste problemen, vilket innehåller:

- Ett initialt tillstånd
- Handlingar utan tidsfördröjning (utförs direkt)
- Deterministiska handlingar (en handling kan aldrig misslyckas och utförs en efter en)
- En enda entitet (fordon eller resenär)

I vårt fall blir det tydligt att världen är mer komplex än så, och enbart definitionen av en enda entitet gör att klassisk planering inte är tillämpbar för oss. För att kunna beskriva lite andra, mer praktiskt tillämpbara scenarion än i fallet klassisk planering har det kommit uppdateringar till PDDL. I PDDL2.1[6] finns stöd för att planera med numeriska flyttal, vilket bland annat är användbart för att uppskatta avstånd, tid och kostnader. Det finns också stöd för durativa handlingar, d.v.s. handlingar som kan ta olika lång tid att utföra, med varierande villkor och följdeffekter. Nu börjar vi närma oss att kunna beskriva det problem med ruttplanering som vi vill lösa i PDDL och låta en planerare hitta en lösning åt oss. Det fattas dock en viktig del, som handlar om att kunna hantera händelser som sker vid en viss tid, utan att det har med planen och göra. I vårt fall betyder det att passagerare bara kan bli hämtade och lämnade inom ett visst tidsintervall. För att kunna beskriva det behövs en uppfattning om tid och när det är tillåtet att hämta respektive lämna en passagerare. Den egenskapen kom med uppdateringen PDDL2.2 och kallas för ‘timed initial literals’[7]. Nästa steg handlar om att hitta en planerare[8] som uppfyller våra krav på domänen och som löser problemet på det sättet vi önskar.

2.1. Planerare

Domänen och problemen vi beskriver kräver att planeraren kan hantera följande krav:

- :durative-actions (PDDL2.1)
- :fluents (PDDL2.1)
- :timed-initial-literals (PDDL2.2)

[6] https://planning.wiki/ref/pddl21
[7] https://planning.wiki/ref/pddl22/requirements#timed-initial-literals
[8] https://planning.wiki/ref/planners

- :duration-inequalities (PDDL2.1)

Det finns också lite olika taktiker och tekniker som planerare använder vid sökning, vilket gör att det finns några olika kategorier av planerare. Vissa planerare försöker exempelvis endast hitta en optimal lösning, medan andra försöker hitta en tillräckligt bra lösning. Det första kallas för 'optimal planners' och det andra för 'satisficing planners'. Sedan finns såklart klassisk planering där handlingar sker direkt i en sekvens, och domäner där handlingar kan ske samtidigt oberoende av varandra. Det senare kallas för 'temporal planning'[9] och är ett krav för att vi skall kunna planera med flera fordon samtidigt. Det är inte heller rimligt att anta att vi skall kunna hitta en optimal lösning på ett så stort problem, vilket gör att vi söker efter en så kallad 'Temporal Satisficing Planner'. Tittar vi på den senaste internationella planerings-tävlingen för sådana planerare, International Planning Competition (IPC)[10] från 2018 hittar finner vi fyra nya planerare plus OPTIC som användes som referens. OPTIC är även en av planerarna som fullt ut stödjer kraven i vår domän[11].

2.2. Liknande tillämpningar

Att planera och schemalägga flera olika fordon för olika uppgifter är inte ett unikt problem och det finns flera tillämpningar där PDDL kan vara till hjälp. Ett exempel som också beskriver problemet med tidsfönster i en artikel från 2014[12]. Ett annat bra exempel på konkreta tillämpningar är en masteruppsats[13] från 2016 som belyser några av de svårigheter som finns med temporal planning. Uppsatsen visar att ett av dom stora problemen, förutom att beskriva själva domänen realistiskt och tolka planerna, är komplexiteten som kommer med temporära domäner. State-of-the-art-planerare löser enklare problem väldigt effektivt, men har mycket svårare att hantera mer komplexa problem som exempelvis använder numeriska och temporära begränsningar. Detta är verkligen något att ta i beaktning och undersöka när vi testar planerare på vårt problem, även om vi använder snäppet modernare planerare än vad uppsatsen kunde göra 2016. Rapporten visar också att problemet verkar i stor utsträckning vara temporära begränsningar (som exempelvis hämta- och lämna-tiderna) snarare än antalet fordon, i vårt fall. En annan viktig slutsats är att det förmodligen är intressant att titta på olika tekniker för att hantera och minska komplexiteten som automatiskt följer med den här typen av problem.

Det finns också en intressant artikel[14] som bland annat visar hur så kallade 'envelope actions' kan användas för att modellera att exempelvis ett fordon enbart kostar pengar så fort det är utanför garaget, vilket är en av grundförutsättningarna för att kunna hitta kostnadseffektiva

[9] https://planning.wiki/_citedpapers/pddl212003.pdf
[10] https://ipc2018-temporal.bitbucket.io/#
[11] https://planning.wiki/ref/planners/optic
[12] https://link.springer.com/content/pdf/10.1007/978-3-662-44980-6_23.pdf
[13] http://www.davidedellanna.com/docs/MSc_Thesis_DellAnna.pdf
[14] http://www.itu.dk/people/rmj/data/techRep/tccj-tr-12.pdf

planer i vårt fall. Nu har vi kommit så pass långt att vi kan börja skriva en domän som tar hänsyn till gällande regelverk och beskriva problemet på ett sådant sätt att vi kan få meningsfulla lösningar från en planerare. Vi tar dessutom i beaktande att ruttplanering är komplext och vi är beredda på att försöka hitta vägar som minskar komplexiteten för att kunna göra några första tester.

3. Metod

Här beskrivs hur vi gick tillväga för att beskriva en domän och utvärdera lösningarna på några problem.

3.1. Geografier

För att kunna konstruera realistiska planer är det viktigt att vi vet både körsträckan och förväntad körtid mellan två adresser. I det system som Östgötatrafiken använder idag är adresser indelade i så kallade *statistik-zoner* som beskriver ett geografiskt område och uppskattad körtid mellan dessa. Det krävs erfarenhet och kunskap om lokala företeelser för att kunna justera dessa tider, vilket det finns ett behov av i dagens system för att få fram realistiska planer.

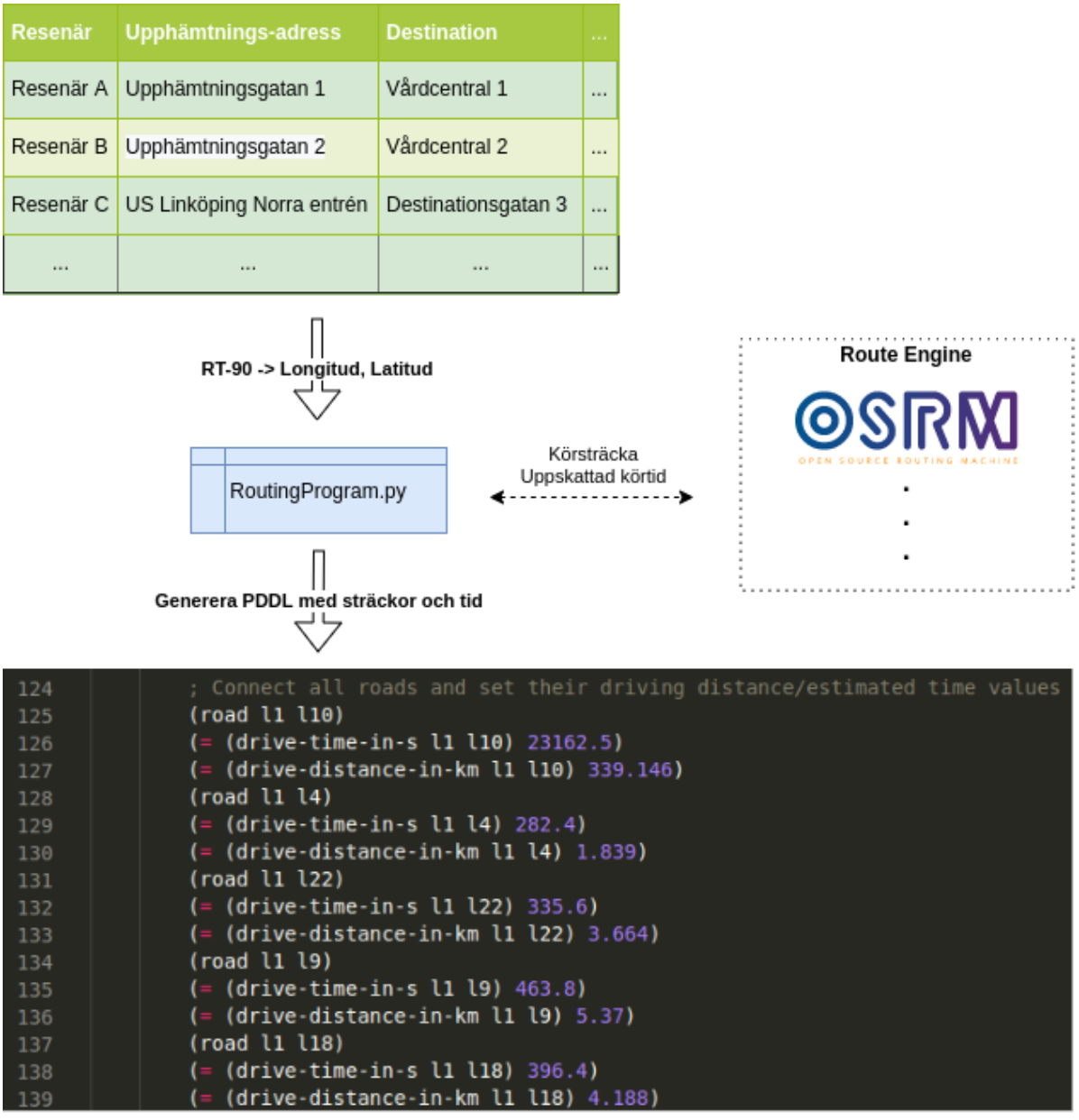
I det här projektet har vi tittat på hur vi kan:

- Hitta mer exakta körsträckor och körtider mellan olika adresser
- Bygga ett system som inte kräver så mycket lokalkännedom och manuell handpåläggning
- Bygga ett dynamiskt system som är enkelt att ändra i och som i framtiden kan ta hänsyn till realtidsinformation från körsträckor.

Det finns en uppsjö av olika motorer för ruttplanering mellan två eller flera platser, som exempelvis Google Maps och motorer för OpenStreetMap[15]. I det här projektet valde vi att testa och använda Open Source Routing Machine (OSRM)[16]. OSRM tillhandahåller ett API där det går att fråga om körsträckan (i meter) och uppskattad restid med bil (i sekunder) mellan två adresser. I det här projektet användes deras Docker-container[17] med en Sverigekarta lokalt. Figuren nedan visar hur OSRM användes för att uppskatta körsträcka och tid mellan två platser och hur resultatet sedan skrivs i PDDL så att en planerare får den informationen. Notera att adresserna i det nuvarande systemet använder koordinatsystemet RT-90[18] och att en konvertering till longitud och latitud gjordes för att kunna använda OSRM. Notera också att det enkelt går att byta ut OSRM mot en eventuell annan ruttplaneringsmotor

[15] https://wiki.openstreetmap.org/wiki/Routing
[16] http://project-osrm.org/
[17] https://hub.docker.com/r/osrm/osrm-backend/
[18] https://www.lantmateriet.se/sv/Kartor-och-geografisk-information/gps-geodesi-och-swepos/Referenssystem/Tvadimensionella-system/RT-90/

om så önskas. För att göra konverteringen, hämta information från OSRM och generera PDDL användes ett litet program som i figuren heter ‘RoutingProgram.py’.




```
124         ; Connect all roads and set their driving distance/estimated time values
125         (road l1 l10)
126         (= (drive-time-in-s l1 l10) 23162.5)
127         (= (drive-distance-in-km l1 l10) 339.146)
128         (road l1 l4)
129         (= (drive-time-in-s l1 l4) 282.4)
130         (= (drive-distance-in-km l1 l4) 1.839)
131         (road l1 l22)
132         (= (drive-time-in-s l1 l22) 335.6)
133         (= (drive-distance-in-km l1 l22) 3.664)
134         (road l1 l9)
135         (= (drive-time-in-s l1 l9) 463.8)
136         (= (drive-distance-in-km l1 l9) 5.37)
137         (road l1 l18)
138         (= (drive-time-in-s l1 l18) 396.4)
139         (= (drive-distance-in-km l1 l18) 4.188)
```

Flödesschema från en bokningslista till PDDL.

3.2. Regelverk

För att kunna bygga en domän som beskriver verkligheten, vad vi kan och inte kan göra behöver vi verkligen förstå regelverket och kraven det ställer på hur planerna får utformas. Listan nedan listar centrala regler i dagens ruttplanering som vi sedan måste adressera i domänen:

- Förskjutningstiden är önskad tid +/- 30 minuter
- Direktresetiden (restid + hanteringstid) + 30 minuter är maximal samplanerad resa
- 2 minuter hanteringstid per resenär vid hämtning och lämning
    - Ingen extra hanteringstid om hjälpmedel rollator finns med
    - Extra tid vid hämtning och lämning om följande hjälpmedel finns med
        - +3 minuter för rullstol eller permobil
        - +2 minuter om extra hjälp finns med
- En rullstol tar upp 1 rullstolsplats i SPEC-buss
- En permobil tar upp 2 rullstolsplatser i SPEC-buss
- Rollator påverkar ej utrymmet i fordonet
- En resenär per sittplats
- Resenär med hjälpmedel rullstol eller permobil tar upp en eller två rullstolsplats(er)
- Resenärer med rullstol eller permobil kan enbart färdas med SPEC-buss

3.3. Trafikavtal

Vi behöver också modellera vad de olika fordonen har för avtal för att göra realistiska planer. Följande tre trafikavtal var aktiva vid tiden för det här projektet:

Avtal 1:
- 34 SPEC-bussar (5 sittande + 2 rullstolsplatser)
- 222 kr/h
- 4.19 kr/km
- Garage: Östra malmskogen i Linköping

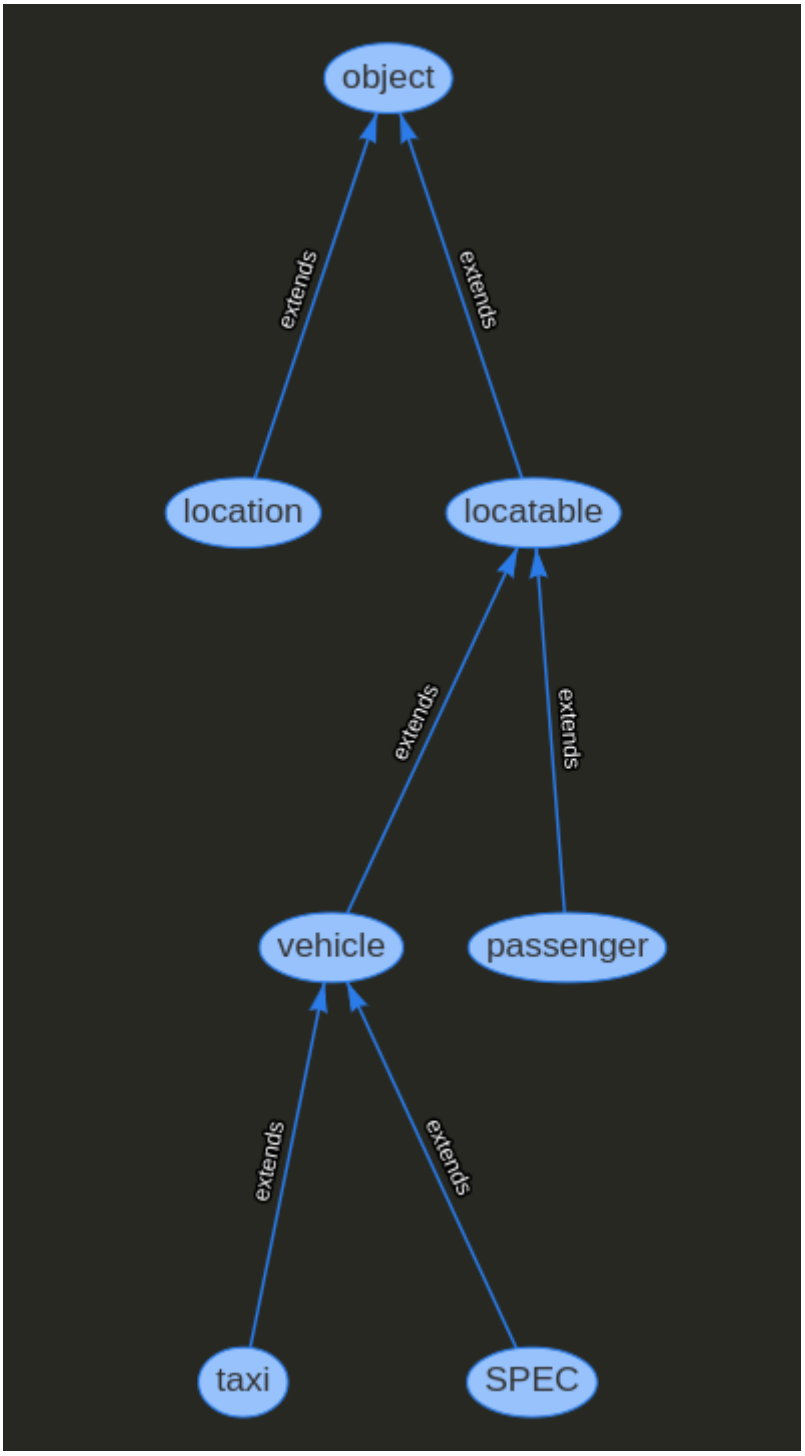
Avtal 2:
- 34 SPEC-bussar (5 sittande + 2 rullstolsplatser)
- 250 kr/h
- 4.00 kr/km
- Garage: Roxengatan i Linköping

Avtal 3:
- 20 taxibilar (4 sittplatser)
- 250 kr/h
- 3,50 kr/km
- Garage: Roxengatan i Linköping

3.4. Domänen

Domänen försöker att i det stora hela beskriva och avspegla det regelverk som finns. Vi specificerar också vilka typer av objekt som finns. Figuren nedan visar vilka typer som används och hur hierarkin mellan typerna ser ut. En ‘locatable’ beskriver något som som kan byta position, det vill säga en resenär eller ett fordon. Ett fordon i sin tur är uppdelat i taxi och SPEC, som är två fordon med lite olika egenskaper. Genom att modellera domänen på det här

sättet är det enkelt att lägga till och ta bort nya typer av fordon och/eller resenärer medan gemensamma egenskaper delas.



Relationen mellan typer i domänen.

I domänen beskriver vi också möjliga handlingar, vad som krävs för att få köra en handling samt vad effekten av handlingen blir. Vi beskriver också att vissa handlingar (som exempelvis att lämna garaget och köra runt) medför en kostnad. Ett exempel på en av handlingarna i

domänen är handlingen ‘drive’ som beskriver hur ett fordon kan flytta mellan två olika platser. Bilden nedan visar hur drive är modellerad i domänen.

```
(:durative-action drive
    :parameters (?v - vehicle ?from ?to - location)
    :duration (= ?duration (drive-time-in-s ?from ?to))
    :condition (and (at start (at ?v ?from))
                    (over all (on-mission ?v))
                    (over all (road ?from ?to)))
    :effect (and
              (at start (and (not (at ?v ?from))
                             (increase (total-cost) (* (drive-distance-in-km ?from ?to) (cost-per-km ?v) ))))
              (at end (and (at ?v ?to)
                           (not (at ?v ?from))
                           (vehicle-has-moved ?v)
                           ))))
```

Handlingen ‘drive’ som flyttar ett fordon mellan två platser.

Till att börja med ser vi att handlingen är en ‘durative-action’, alltså en handling som sträcker sig över ett tidsintervall. Handlingar tar tre parametrar. En av typen fordon och två av typen plats. Det vi behöver för att utföra handlingen ‘drive’ är således ett fordon, en plats att köra från och en plats att köra till. Kraven för att få utföra handlingen är att fordonet befinner sig på platsen ‘from’, att fordonet har fått ett uppdrag att göra en körning samt att det finns en planerad rutt mellan platserna ‘from’ och ‘to’.

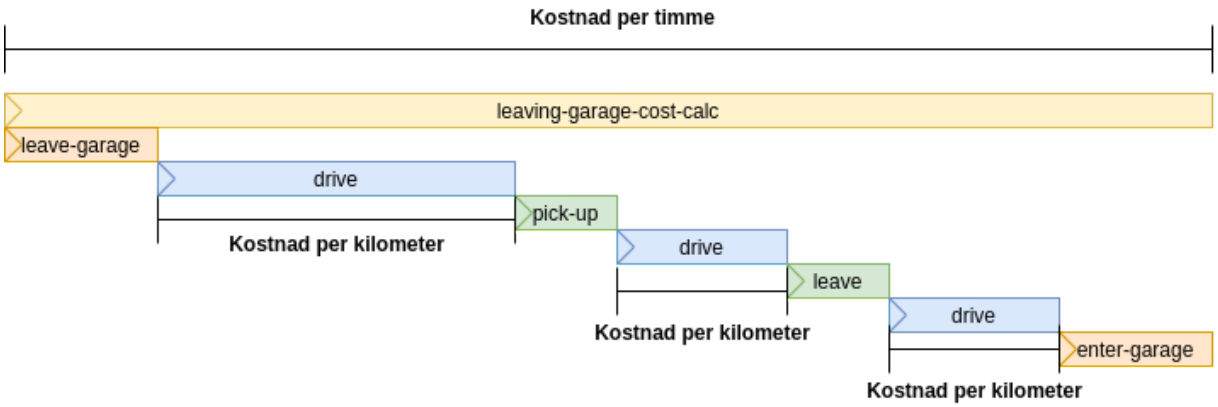
Effekten av att utföra handlingen blir att den totala kostnaden för hela planen ökar med kostnaden per kilometer multiplicerat med distansen mellan platserna, som vi fått från vår geografi. Vi kan också se att fordonet hamnar på platsen ‘to’ istället för ‘from’ samt att fordonet har flyttat på sig när handlingen har utförts.

Det finns också handlingar för att hämta och lämna resenärer. Dessa påminner om varandra och i exemplet nedan är det handlingen ‘pick-up-passenger’ som beskrivs. Givet ett fordon, en plats och en resenär samt att fordonet och resenären är på samma plats, passageraren är redo att bli hämtad (tid på dygnet), att fordonet är i trafik och att föraren kan hjälpa till kan handlingen utföras. Den beräknade tidsåtgången för handlingen uppskattas till hanteringstiden för passageraren som hämtas, vilket bestäms redan vid legitimeringen.

```
(:durative-action pick-up-passenger
    :parameters (?v - vehicle ?current - location ?p - passenger)
    :duration (= ?duration (handling-time ?p))
    :condition (and (at start (at ?p ?current))
                    (over all (ready-for-pickup ?p))
                    (over all (at ?v ?current))
                    (over all (on-mission ?v))
                    (at start (driver-available ?v)))
    :effect (and
                (at start (and (not (at ?p ?current))
                               (not (driver-available ?v))))
                (at end (and (in ?p ?v)
                             (driver-available ?v)))))
```

Handlingen 'pick-up-passenger' som hämtar en resenär med hänsyn till resenärens hanteringstid.

Eftersom att kostnaden för en ruttplanering baseras sig på fordonens körsträcka och körtid behöver vi ett sätt att modellera tid i domänen, för att sedan kunna låta planeraren söka efter planber som minimerar den totala kostnaden. Detta går att lösa genom en 'envelope action', vilket är en handling som utförs på en, initialt obestämd tid genom att omfamna en eller flera verkliga handlingar i syfte att beräkna och minimera kostnaden. I vårt fall kostar ett fordon pengar så fort ett fordon lämnar garaget. När fordonet sedan är ute och kör kan flera olika handlingar utföras, som exempelvis att åka och hämta och lämna passagerare. När fordonet sedan återvänder till garaget är vår 'envelope action' slut, och vi kan veta hur länge fordonet beräknas vara i trafik, vilket gör att vi kan uppskatta och minimera kostnaden för sejouren. Figuren nedan visar ett exempel när ett fordon lämnar garaget, åker från garaget till en resenär, kör till resenärens destination och sedan återvänder till garaget.



En sekvens av handlingar som går att optimera på kostnad med avseende på sträcka och tid.

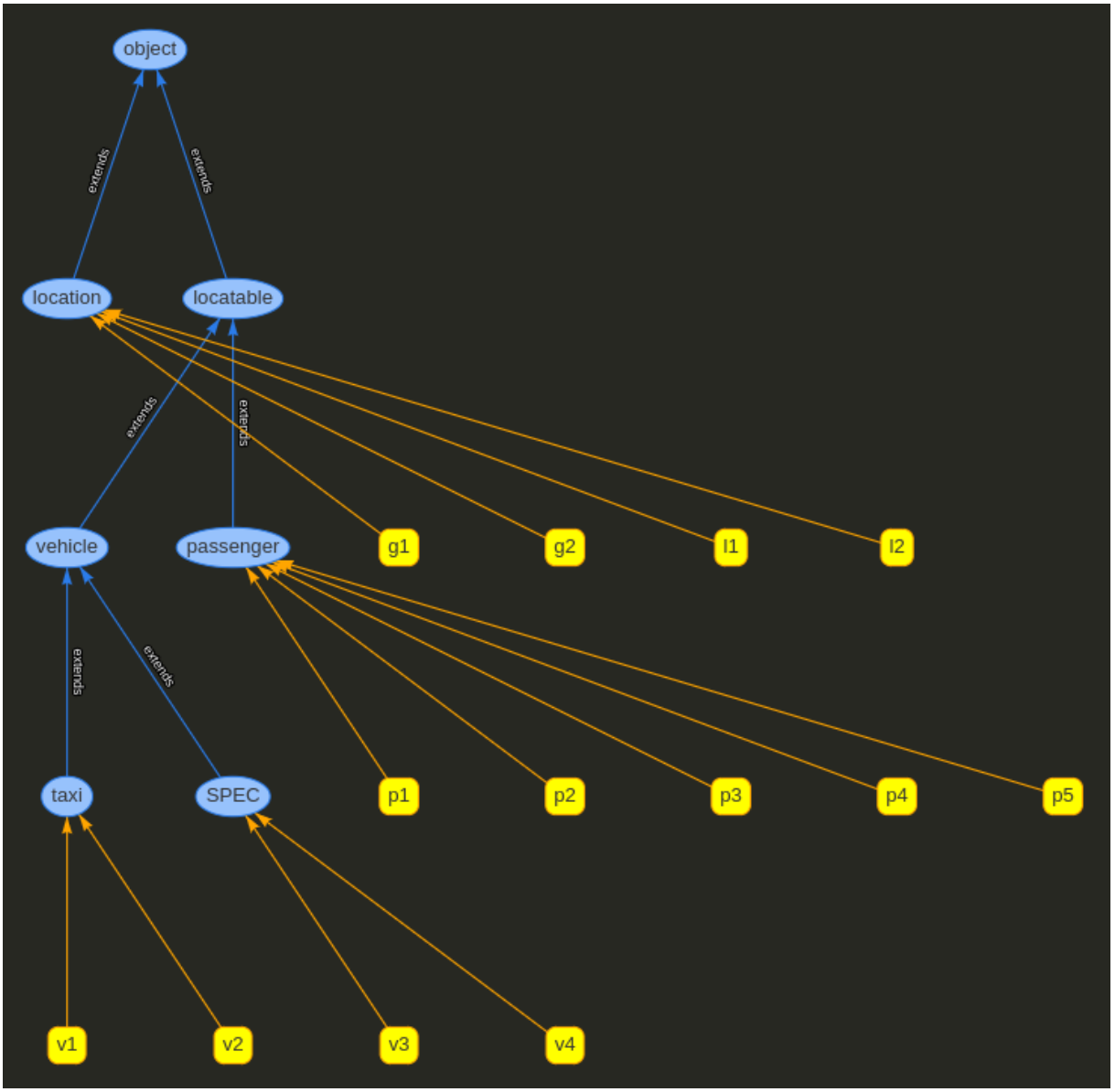
Bilden nedan visar handlingen 'leaving-garage-cost-calc' som används för att beskriva att fordon kostar pengar varje gång det lämnar garaget, i enlighet med trafikavtalen i regelverket.

```
(:durative-action leaving-garage-cost-calc
    :parameters (?v - vehicle)
    :duration (and (>= ?duration 0.01) (<= ?duration 10000))
    :condition (and
        (at start (cost-calc-mutex ?v))
        (at start (start-vehicle-cost ?v))
        (at end (stop-vehicle-cost ?v))
    )
    :effect (and
        (at start (and
            (not (cost-calc-mutex ?v))
            (not (start-vehicle-cost ?v))
            (allowed-to-leave-garage ?v)
            (increase (total-cost) (* (cost-per-second ?v) ?duration))
        ))
        (at end (and
            (driving-cost-calculated ?v)
            (cost-calc-mutex ?v)
            (start-vehicle-cost ?v)
            (not (allowed-to-leave-garage ?v))
            (not (stop-vehicle-cost ?v))
        ))
    )
)
```

Handlingen 'leaving-garage-cost-calc' som måste utföras för att ett fordon skall få lämna garaget. Avslutar när fordonet återvänder till garage.

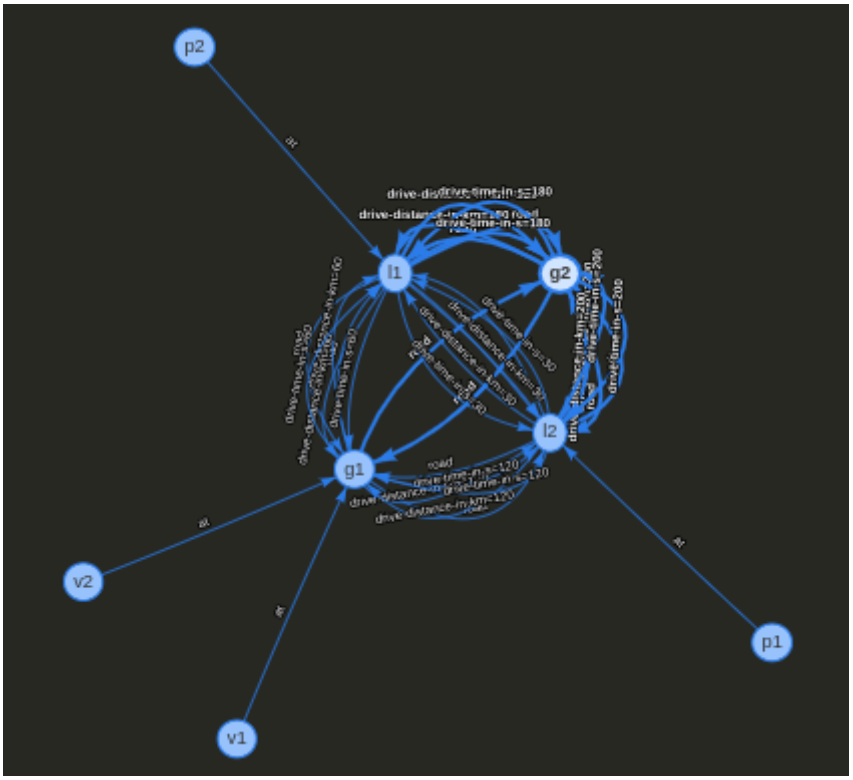
3.5. Problem

Problemfilen instansierar objekten och talar exempelvis om exakt hur många fordon det finns, vart dom är och hur många som kan åka med. Till att börja med testade vi väldigt små domänen för att modellera och bygga konceptet. Bilden nedan visar ett problem där vi har två stycken taxibilar och två SPEC-bussar att tillgå. Det finns liksom i verkligheten två garage som bilarna initialt befinner sig på. Sedan har vi i det här exemplet två passagerare/resenärer och två olika platser utöver garagen. Som tidigare nämnt är det väldigt enkelt att forma problemet efter hur verkligheten ser ut just för tillfället.



Objekten i ett litet test-problem för vår domän.

Nästa bild visar på hur det initiala tillståndet i det här mindre problemet ser ut. Vi kan se att passagerare ‘p2’ befinner sig på position ‘l1’ och passagerare ‘p1’ på ‘l2’. Vi kan också se att alla vägar och garage är sammankopplade med en väg och att det tar lite olika tid att åka mellan dem. Det framgår också att vi endast har de två taxibilarna ‘v1’ och ‘v2’ att tillgå och att dem befinner sig i garage ‘g1’. Det är också i det initiala tillståndet som vi, med hjälp av geografin, uppskattat sträckan och tiden mellan aktuella platser. Vi specificerar dessutom vilket avtal varje fordon kör under för att på så sätt ge planeraren chansen att välja fordon som kostar minst för den uppgift som skall lösas.



Grafisk beskrivning av det initiala tillståndet i ett mindre delproblem.

Det vi nu försöker åstadkomma är att hitta en sekvens av handlingar som transformerar grafen till måltillståndet och som samtidigt försöker att minimera den totala kostnaden. Det uttrycker vi också i vår problemfil som vi återigen enkelt kan uppdatera efter våra önskemål. Bilden nedan visar hur det ser ut i PDDL om vi vill att passagerarna exempelvis skall byta plats med varandra.

```
(:goal (and (at p1 l1)
            (at p2 l2)
))

(:metric minimize (total-cost))
```

Grafisk beskrivning av det initiala tillståndet i ett mindre delproblem.

Nu vet vi hur vi modellerar vårt initiala tillstånd, vårt måltillstånd samt hur vi kan beskriva handlingar för planeraren som transformerar vårt initiala tillstånd till vårt mål. En av de sista och viktigaste komponenterna att modellera för att beskriva ett verkligt problem är tidsaspekten. Som regelverket beskriver finns det en förskjutningstid på +/- 30 minuter samt en maximal samplaneringstid på 30 minuter per resenär. Det innebär att om en resenär bokar en resa för upphämtning till 12:00 får resenären räkna med att bli upphämtad någon gång mellan 11.30 och 12.30. Detta är självklart en central del i en ruttplanering och att vi tar hänsyn till det när vi planerar. Vi beskriver den här tidsfristen i problemfilen genom att

specificera ett tidsfönster som beskriver när resenären är redo att bli hämtad. Det innebär att ‘pick-up-passenger’ enbart fungerar under tiden som ‘ready-for-pickup’ är sant, vilket är just önskad avgångstid +/- 30 minuter. Detta framgår av bilden nedan, där passagerare ‘p1’ vill bli hämtad 12:00 och passagerare ‘p2’ 14:00. Tiden är uttryckt i sekunder efter midnatt.

```
; List of booked passangers
(at p1 l2)
(= (handling-time p1) 120)
(at 41400 (ready-for-pickup p1))
(at 45000 (not (ready-for-pickup p1)))

(at p2 l1)
(= (handling-time p2) 120)
(at 48600 (ready-for-pickup p2))
(at 52200 (not (ready-for-pickup p2)))
```

Lista med egenskaper för varje passagerare som är inbokad. Här går det att lägga på fler krav som exempelvis behov av rullstol eller extra visad hänsyn.

Vi har nu lyckats uttrycka alla centrala delar av vårt problem med hjälp av tillägg till PDDL. Resultatet kommer att beskriva några tester som genomfördes och de planer som skapades. Sedan förs en vidare diskussion om vad resultatet innebär för framtida arbete och några intressanta tekniker för att kunna minska komplexiteten.

3.6. Instruktioner för att bygga och köra planeraren OPTIC för Ubuntu 18.04

En stor del i det här projektet var att kunna bygga och köra planerare som uppfyller kraven som vår domän ställer. Instruktionerna beskriver lärdomar från att planera med OPTIC, som användes som riktmärke av andra planerare i den senaste internationella planeringstävlingen 2018[19] för ‘temporal tracks’.

Dokumentationen för tävlingen föreslår att OPTIC (och övriga planerare i tävlingen) byggs med hjälp av en Singularity-container[20], vilket kan vara bra att börja testa med. Dessvärre är inga versioner av paketen som använts specificerade och många planerare använder exempelvis äldre versioner av flex och kommer inte att fungera utan att manuellt försöka installera de versioner som använts vid utveckling av just den planeraren. Dessutom försöker Singularity-scriptet bygga alla versioner av OPTIC (med både optimeraren CPLEX[21] och CLP[22]). Ett enkelt första steg är att istället börja med att använda CLP och bygga OPTIC mot

[19] https://ipc2018-temporal.bitbucket.io/
[20] https://singularity.lbl.gov/
[21] https://www.ibm.com/analytics/cplex-optimizer
[22] https://launchpad.net/ubuntu/+source/clp

den optimeraren enligt instruktionerna[23]. Notera att för att kunna göra abstraktioner av 'timed initial literals' så behöver CPLEX användas. För att få det att fungera krävs det att sökvägarna till det biblioteket stämmer överens med dem i exempelvis 'run-cmake-for-cplex-static-x86'. Här krävs det förmodligen att CPLEX först installeras, och att sökvägarna uppdateras eller att biblioteks-filerna flyttas.
Störst framgång med att installera OPTIC nåddes genom att använda scripten make i '*src/optic'* och sedan bygga önskad version av OPTIC med hjälp av Makefilen i '*src/optic/src/Makefile*'. För att exempelvis bygga OPTIC mot CLP användes:

```
make clp
```

Alla tillgängliga alternativ för att bygga OPTIC går att få fram med:

```
make help
```

För att sedan köra exempelvis OPTIC med CPLEX med TIL-abstraktion används följande kommando:

```
./optic-cplex -0 domain.pddl problem.pddl plan_output.txt
```

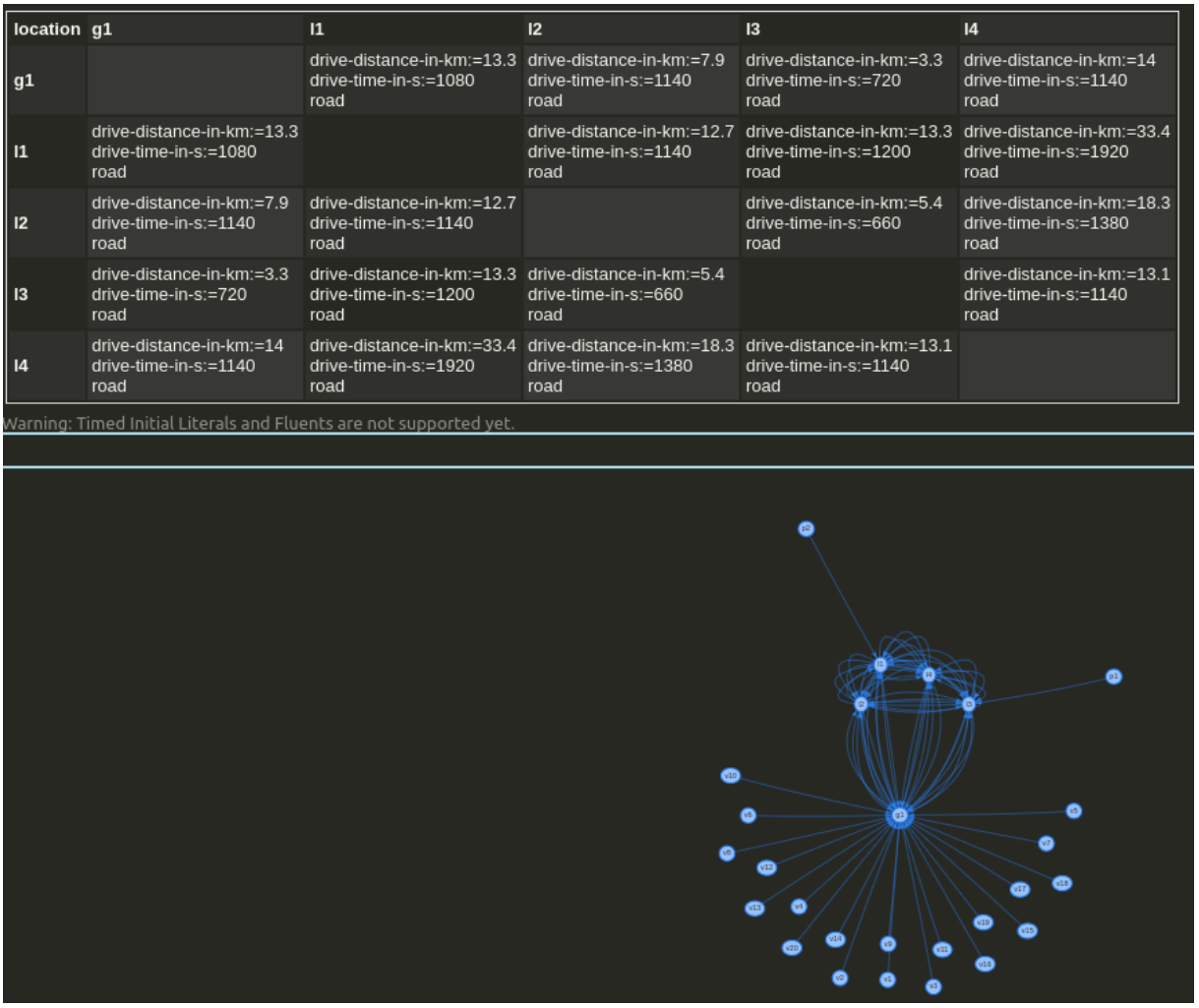
4. Resultat

Resultatet beror i stor utsträckning på hur planeraren är konstruerad, och i det här projektet var det enbart OPTIC som vi lyckades få att fungera fullt ut. En annan planerare som hade varit intressant att testa och jämföra med i framtiden är SMTPlan[24]. Därför är tyvärr resultaten lite begränsade till OPTIC:s förmåga att söka planer i just den här ganska krävande domänen.

Vi börjar utvärdera ett småskaligt problem, problem A, som baserar sig på två resenärer från verkligheten. Av säkerhetsskäl kommer inte dom exakta resorna att redovisas. Det intressanta är att titta på relationerna mellan platserna och hur lösningen ser ut.

Problemet består av två resenärer, där den ena vill resa 12:00 och den andra 12:45. Vi har för enkelhetens skull enbart det tredje trafikavtalet aktiverat, med 20 taxibilar tillgängliga. Bilden nedan visar det initiala tillståndet, där alla taxibilar står i garaget på Roxengatan i Linköping och de två resenärerna på två olika platser i Östergötland. Vi ser också hur vägarna är anslutna och vad geografins uppskattningar är för körsträcka och tid. Målet är att passagerarna skall befinna sig på sina respektive destinationer.

[23] https://nms.kcl.ac.uk/planning/software/optic.html
[24] https://github.com/KCL-Planning/SMTPlan



location	g1	l1	l2	l3	l4
g1		drive-distance-in-km:=13.3 drive-time-in-s:=1080 road	drive-distance-in-km:=7.9 drive-time-in-s:=1140 road	drive-distance-in-km:=3.3 drive-time-in-s:=720 road	drive-distance-in-km:=14 drive-time-in-s:=1140 road
l1	drive-distance-in-km:=13.3 drive-time-in-s:=1080 road		drive-distance-in-km:=12.7 drive-time-in-s:=1140 road	drive-distance-in-km:=13.3 drive-time-in-s:=1200 road	drive-distance-in-km:=33.4 drive-time-in-s:=1920 road
l2	drive-distance-in-km:=7.9 drive-time-in-s:=1140 road	drive-distance-in-km:=12.7 drive-time-in-s:=1140 road		drive-distance-in-km:=5.4 drive-time-in-s:=660 road	drive-distance-in-km:=18.3 drive-time-in-s:=1380 road
l3	drive-distance-in-km:=3.3 drive-time-in-s:=720 road	drive-distance-in-km:=13.3 drive-time-in-s:=1200 road	drive-distance-in-km:=5.4 drive-time-in-s:=660 road		drive-distance-in-km:=13.1 drive-time-in-s:=1140 road
l4	drive-distance-in-km:=14 drive-time-in-s:=1140 road	drive-distance-in-km:=33.4 drive-time-in-s:=1920 road	drive-distance-in-km:=18.3 drive-time-in-s:=1380 road	drive-distance-in-km:=13.1 drive-time-in-s:=1140 road	

Warning: Timed Initial Literals and Fluents are not supported yet.

Initialt tillstånd för det mindre exemplet.

Bilden nedan visar den plan som följande kommando gav:

```
./optic-cplex -0 domain.pddl problem.pddl plan_output.txt
```

Vi lyckades alltså hitta en plan, som kostar 697 kronor att köra och som använder två olika fordon, på 0.44 sekunder! Tidsstämpeln till vänster visar när handlingen utförs (i sekunder från midnatt). Vi ser hur en av taxibilarna lämnar garaget 1080 sekunder innan passagerare ‘p2’ får bli upphämtad som tidigast, vilket är 30 minuter innan önskad resa, alltså 11:30. Vi ser också att det är planerat 120 sekunder för att hjälpa resenären ombord, vilket framgår av klammern [120.000] till höger om ‘(pick-up-passenger v1 l1 p2)’. Vi ser också att den andra taxibilen gör en liknande resa mot den andra passageraren, hämtar, kör och lämnar innan resan går tillbaka mot garaget. Vi lyckades alltså lösa vårt lilla problem på bråkdelen av en sekund!

```
; Plan found with metric 697.012
; Theoretical reachable cost 697.012
; States evaluated so far: 124
; States pruned based on pre-heuristic cost lower bound: 0
; Time 0.44
43019.991: (leaving-garage-cost-calc v1)  [3600.012]
43019.991: (leave-garage v1 g1)  [0.010]
43020.001: (drive v1 g1 l1)  [1080.000]
44100.001: (pick-up-passenger v1 l1 p2)  [120.000]
44220.001: (drive v1 l1 l2)  [1140.000]
44279.991: (leaving-garage-cost-calc v10)  [3240.012]
44279.991: (leave-garage v10 g1)  [0.010]
44280.001: (drive v10 g1 l3)  [720.000]
45000.001: (pick-up-passenger v10 l3 p1)  [120.000]
45120.001: (drive v10 l3 l4)  [1140.000]
45360.001: (leave-passenger v1 l2 p2)  [120.000]
45480.001: (drive v1 l2 g1)  [1140.000]
46260.001: (leave-passenger v10 l4 p1)  [120.000]
46380.001: (drive v10 l4 g1)  [1140.000]
46620.001: (enter-garage v1 g1)  [0.010]
47520.001: (enter-garage v10 g1)  [0.010]
```

Lösningen på exemplet ovan.

Nu när vi vet att vi kan modellera domänen och hitta en lösning för ett mindre problem blir det högintressant att öka komplexiteten och röra oss mot hur det faktiskt ser ut i verkligheten med planer som täcker några hundra resenärer. Därför utökar vi nu antalet platser och resenärer i problem B.

I problem B använder vi samma domän, men utökar antalet platser till 25 stycken och antalet resenärer till 5. Resultatet blir att vi ökar antalet möjliga körvägar enligt följande ekvation, där $n$beskriver antalet platser:

$$n(n - 1)$$

Eftersom att vi måste beskriva en väg mellan två platser två gånger (A till B och B till A) ger det oss 600 vägar att ta hänsyn till ihop med komplexiteten som följer med passagerarnas tidsfönster. Detta gör att vi idag inte kan lösa ruttplaneringen under en tillfredsställande tid (30 minuter) och vi lämnar till vidare arbete att undersöka möjligheter för att få detta att fungera tillfredsställande.

5. Diskussion

Vi har visat att det går att komma långt med automatisk planering, om än inte hela vägen för att lösa hela problemet i det här projektet med tanke på problem B. Vi kan uttrycka problemet

med hjälp av PDDL och på så sätt få hjälp av en state-of-the-art-planerare för att lösa och optimera planen med avseende på sträcka och tid. Vi har också visat att det finns alternativa ruttplaneringsmotorer och geografier som i framtiden kan ta hänsyn till och inkorporera aktuell status mellan två platser som exempelvis trafiktäthet, olyckor och vägarbeten, i den slutgiltiga planen.

Det behövs tekniker som hjälper planeraren att hålla nere komplexiteten för att kunna komma fram till en plan inom en rimlig tid. En av dom mest lovande teknikerna är att minska antalet väg-kopplingar på ett smart sätt. Det skulle exempelvis kunna vara att varje plats är sammankopplad med dom *k* närmaste platserna, plus garagen och destinationen. På så sätt minskar antalet koppling kraftigt och problemet skalar mycket bättre. Variabeln *k* innebär också att det finns möjlighet att hitta vägar för samåkning. Det kommer dock att kräva lite extra handpåläggning för att förfina planen efteråt.

6. Slutsatser

- Det finns redan idag planerare som stödjer funktioner för att kunna modellera en domän ruttplanering.
- Det finns nya/andra planerare som vore intressanta att testa.
- Fältet är under ständig utveckling vilket är lovande för exempelvis nya heuristiker.
- Det är extremt viktigt att kunna hålla nere komplexiteten och antalet ‘states’ för att kunna hitta en plan inom rimlig tid.
- Vi kan med enkelhet hitta planer för problem som liknar problem A.
- Vi hittar inte en lösning på problem i likhet med problem B, inom en rimlig tid.
- Automatisk planering separerar bokningar och ruttplanering på ett föredömligt sätt.
- Det finns olika geografier och ruttplaneringsmotorer plats-till-plats att utvärdera.
- Automatiskt tar hänsyn till snabba förändringar i kombination med kostnadseffektiva planer (samplanering).
- Samplanering är ett fantastiskt sätt att minska kostnader.

7. Appendix A

7.1. Domänen

```
; Domain for regional medical transportation in Östergötland

(define (domain transportation_service)
   (:requirements :typing :adl :durative-actions :fluents
:timed-initial-literals :duration-inequalities)
   (:types location locatable - object
           vehicle passenger - locatable
```


```
            taxi - vehicle
            SPEC - vehicle)
    (:predicates
        (at ?v - locatable ?x - location)
        (in ?p - passenger ?v - vehicle)
        (road ?x ?y - location) ; There is a road between two locations
        (ready-for-pickup ?p - passenger) ; A passenger is ready to be
picked-up
        (on-mission ?v - vehicle)
        (driver-available ?v - vehicle)
        (leave-passenger-mutex ?v - vehicle)
        (driving-cost-calculated ?v - vehicle)
        (cost-calc-mutex ?v - vehicle)
        (garage ?l - location) ; This location is a garage
        (stop-vehicle-cost ?v - vehicle)
        (start-vehicle-cost ?v - vehicle)
        (allowed-to-leave-garage ?v - vehicle)
        (vehicle-has-moved ?v - vehicle)
        (empty ?v - vehicle)
    )
    (:functions
        (drive-time-in-s ?from ?to - location); Estimated drive time
between two locations (seconds)
        (drive-distance-in-km ?from ?to - location); Estimated driving
distance between two locations (km)
        (drive-cost ?v - vehicle ?from ?to - location); Estimated drive
cost between two locations
        (handling-time ?p - passenger)
        (total-cost) - number
        (cost-per-second ?v - vehicle)
        (cost-per-km ?v - vehicle)
    )

    (:durative-action leaving-garage-cost-calc
        :parameters (?v - vehicle)
        :duration (and (>= ?duration 0.01) (<= ?duration 10000))
        :condition (and
            (at start (cost-calc-mutex ?v))
            (at start (start-vehicle-cost ?v))
```

```
            (at end (stop-vehicle-cost ?v))
        )
        :effect (and
            (at start (and
                (not (cost-calc-mutex ?v))
                (not (start-vehicle-cost ?v))
                (allowed-to-leave-garage ?v)
                (increase (total-cost) (* (cost-per-second ?v)
?duration))
            ))
            (at end (and
                (driving-cost-calculated ?v)
                (cost-calc-mutex ?v)
                (start-vehicle-cost ?v)
                (not (allowed-to-leave-garage ?v))
                (not (stop-vehicle-cost ?v))
            ))
        )
    )

    (:durative-action leave-garage
        :parameters (?v - vehicle ?l - location)
        :duration (= ?duration 0.01)
        :condition (and
            (over all (and
                (at ?v ?l)
                (garage ?l)
                (allowed-to-leave-garage ?v)
            ))
        )
        :effect (and
            (at end (and
                (on-mission ?v)
            ))
        )
    )

    (:durative-action enter-garage
        :parameters (?v - vehicle ?l - location)
```


```
        :duration (= ?duration 0.01)
        :condition (and
            (over all (and
                (at ?v ?l)
                (garage ?l)
                (on-mission ?v)
                (vehicle-has-moved ?v)
                (empty ?v)
            ))
        )
        :effect (and
            (at start (and
                (stop-vehicle-cost ?v)
            ))
            (at end (and
                (not (on-mission ?v))
                (not (vehicle-has-moved ?v))
            ))
        )
    )

    (:durative-action drive
        :parameters (?v - vehicle ?from ?to - location)
        :duration (= ?duration (drive-time-in-s ?from ?to))
        :condition (and (at start (at ?v ?from))
                        (over all (on-mission ?v))
                        (over all (road ?from ?to)))
        :effect (and
                    (at start (and (not (at ?v ?from))
                                   (increase (total-cost) (*
(drive-distance-in-km ?from ?to) (cost-per-km ?v) ))))
                    (at end (and (at ?v ?to)
                                 (not (at ?v ?from))
                                 (vehicle-has-moved ?v)
                                 ))))

    (:durative-action pick-up-passenger
        :parameters (?v - vehicle ?current - location ?p - passenger)
        :duration (= ?duration (handling-time ?p))
```

```
        :condition (and (at start (at ?p ?current))
                        (over all (ready-for-pickup ?p))
                        (over all (at ?v ?current))
                        (over all (on-mission ?v))
                        (over all (driver-available ?v)))
        :effect (and
                    (at start (and (not (at ?p ?current))
                                   (not (driver-available ?v))))
                    (at end (and (in ?p ?v)
                                 (not (empty ?v))
                                 (driver-available ?v)))))

    (:durative-action leave-passenger
        :parameters (?v - vehicle ?current - location ?p - passenger)
        :duration (= ?duration (handling-time ?p))
        :condition (and (over all (in ?p ?v))
                        (over all (at ?v ?current))
                        (over all (on-mission ?v))
                        (at start (leave-passenger-mutex ?v)))
        :effect (and
                    (at start (not (leave-passenger-mutex ?v)))
                    (at end (at ?p ?current))
                    (at end (not (in ?p ?v)))
                    (at end (empty ?v))
                    (at end (leave-passenger-mutex ?v))))
)
```

7.2. Problem A

```
(define (problem first_problem)
    (:domain transportation_service)
    (:objects p1 p2 - passenger
              v1 v2 v3 v4 v5 v6 v7 v8 v9 v10 v11 v12 v13 v14 v15 v16 v17
v18 v19 v20 - taxi
              g1 l1 l2 l3 l4 - location)
    (:init
        (cost-calc-mutex v1)
```


```
        (cost-calc-mutex v2)
        (cost-calc-mutex v3)
        (cost-calc-mutex v4)
        (cost-calc-mutex v5)
        (cost-calc-mutex v6)
        (cost-calc-mutex v7)
        (cost-calc-mutex v8)
        (cost-calc-mutex v9)
        (cost-calc-mutex v10)
        (cost-calc-mutex v11)
        (cost-calc-mutex v12)
        (cost-calc-mutex v13)
        (cost-calc-mutex v14)
        (cost-calc-mutex v15)
        (cost-calc-mutex v16)
        (cost-calc-mutex v17)
        (cost-calc-mutex v18)
        (cost-calc-mutex v19)
        (cost-calc-mutex v20)

        (start-vehicle-cost v1)
        (start-vehicle-cost v2)
        (start-vehicle-cost v3)
        (start-vehicle-cost v4)
        (start-vehicle-cost v5)
        (start-vehicle-cost v6)
        (start-vehicle-cost v7)
        (start-vehicle-cost v8)
        (start-vehicle-cost v9)
        (start-vehicle-cost v10)
        (start-vehicle-cost v11)
        (start-vehicle-cost v12)
        (start-vehicle-cost v13)
        (start-vehicle-cost v14)
        (start-vehicle-cost v15)
        (start-vehicle-cost v16)
        (start-vehicle-cost v17)
        (start-vehicle-cost v18)
        (start-vehicle-cost v19)
```


```
    (start-vehicle-cost v20)

    (leave-passenger-mutex v1)
    (leave-passenger-mutex v2)
    (leave-passenger-mutex v3)
    (leave-passenger-mutex v4)
    (leave-passenger-mutex v5)
    (leave-passenger-mutex v6)
    (leave-passenger-mutex v7)
    (leave-passenger-mutex v8)
    (leave-passenger-mutex v9)
    (leave-passenger-mutex v10)
    (leave-passenger-mutex v11)
    (leave-passenger-mutex v12)
    (leave-passenger-mutex v13)
    (leave-passenger-mutex v14)
    (leave-passenger-mutex v15)
    (leave-passenger-mutex v16)
    (leave-passenger-mutex v17)
    (leave-passenger-mutex v18)
    (leave-passenger-mutex v19)
    (leave-passenger-mutex v20)

    ; Initialize the total cost of the plan
    (= (total-cost) 0)

    ; Initialize infrastructure
    (garage g1)

    ; Initialize vehicle positions
    (at v1 g1)
    (= (cost-per-second v1) 0.069)
    (= (cost-per-km v1) 3.5)
    (driver-available v1)

    (at v2 g1)
    (= (cost-per-second v2) 0.069)
    (= (cost-per-km v2) 3.5)
    (driver-available v2)
```


```
(at v3 g1)
(= (cost-per-second v3) 0.069)
(= (cost-per-km v3) 3.5)
(driver-available v3)

(at v4 g1)
(= (cost-per-second v4) 0.069)
(= (cost-per-km v4) 3.5)
(driver-available v4)

(at v5 g1)
(= (cost-per-second v5) 0.069)
(= (cost-per-km v5) 3.5)
(driver-available v5)

(at v6 g1)
(= (cost-per-second v6) 0.069)
(= (cost-per-km v6) 3.5)
(driver-available v6)

(at v7 g1)
(= (cost-per-second v7) 0.069)
(= (cost-per-km v7) 3.5)
(driver-available v7)

(at v8 g1)
(= (cost-per-second v8) 0.069)
(= (cost-per-km v8) 3.5)
(driver-available v8)

(at v9 g1)
(= (cost-per-second v9) 0.069)
(= (cost-per-km v9) 3.5)
(driver-available v9)

(at v10 g1)
(= (cost-per-second v10) 0.069)
(= (cost-per-km v10) 3.5)
```


```
        (driver-available v10)

        (at v11 g1)
        (= (cost-per-second v11) 0.069)
        (= (cost-per-km v11) 3.5)
        (driver-available v11)

        (at v12 g1)
        (= (cost-per-second v12) 0.069)
        (= (cost-per-km v12) 3.5)
        (driver-available v12)

        (at v13 g1)
        (= (cost-per-second v13) 0.069)
        (= (cost-per-km v13) 3.5)
        (driver-available v13)

        (at v14 g1)
        (= (cost-per-second v14) 0.069)
        (= (cost-per-km v14) 3.5)
        (driver-available v14)

        (at v15 g1)
        (= (cost-per-second v15) 0.069)
        (= (cost-per-km v15) 3.5)
        (driver-available v15)

        (at v16 g1)
        (= (cost-per-second v16) 0.069)
        (= (cost-per-km v16) 3.5)
        (driver-available v16)

        (at v17 g1)
        (= (cost-per-second v17) 0.069)
        (= (cost-per-km v17) 3.5)
        (driver-available v17)

        (at v18 g1)
        (= (cost-per-second v18) 0.069)
```


```
        (= (cost-per-km v18) 3.5)
        (driver-available v18)

        (at v19 g1)
        (= (cost-per-second v19) 0.069)
        (= (cost-per-km v19) 3.5)
        (driver-available v19)

        (at v20 g1)
        (= (cost-per-second v20) 0.069)
        (= (cost-per-km v20) 3.5)
        (driver-available v20)

        ; List of booked passangers
        (at p1 l3)
        (= (handling-time p1) 120)
        (at 45000 (ready-for-pickup p1))
        (at 48600 (not (ready-for-pickup p1)))

        (at p2 l1)
        (= (handling-time p2) 120)
        (at 44100 (ready-for-pickup p2))
        (at 47700 (not (ready-for-pickup p2)))

        ; Connect all roads...
        (road g1 l1)
        (road g1 l2)
        (road g1 l3)
        (road g1 l4)

        (road l1 g1)
        (road l1 l2)
        (road l1 l3)
        (road l1 l4)

        (road l2 g1)
        (road l2 l1)
        (road l2 l3)
        (road l2 l4)
```


```
        (road l3 g1)
        (road l3 l1)
        (road l3 l2)
        (road l3 l4)

        (road l4 g1)
        (road l4 l1)
        (road l4 l2)
        (road l4 l3)

        ;...and assign a cost calculated by distance (in kilometers) and
travel time (in seconds)
        (= (drive-distance-in-km g1 l1) 13.3)
        (= (drive-distance-in-km g1 l2) 7.9)
        (= (drive-distance-in-km g1 l3) 3.3)
        (= (drive-distance-in-km g1 l4) 14.0)
        (= (drive-distance-in-km l1 l2) 12.7)
        (= (drive-distance-in-km l1 l3) 13.3)
        (= (drive-distance-in-km l1 l4) 33.4)
        (= (drive-distance-in-km l2 l3) 5.4)
        (= (drive-distance-in-km l2 l4) 18.3)
        (= (drive-distance-in-km l3 l4) 13.1)

        (= (drive-distance-in-km l1 g1) 13.3)
        (= (drive-distance-in-km l2 g1) 7.9)
        (= (drive-distance-in-km l3 g1) 3.3)
        (= (drive-distance-in-km l4 g1) 14.0)
        (= (drive-distance-in-km l2 l1) 12.7)
        (= (drive-distance-in-km l3 l1) 13.3)
        (= (drive-distance-in-km l4 l1) 33.4)
        (= (drive-distance-in-km l3 l2) 5.4)
        (= (drive-distance-in-km l4 l2) 18.3)
        (= (drive-distance-in-km l4 l3) 13.1)

        (= (drive-time-in-s g1 l1) 1080)
        (= (drive-time-in-s g1 l2) 1140)
        (= (drive-time-in-s g1 l3) 720)
        (= (drive-time-in-s g1 l4) 1140)
```


```
        (= (drive-time-in-s l1 l2) 1140)
        (= (drive-time-in-s l1 l3) 1200)
        (= (drive-time-in-s l1 l4) 1920)
        (= (drive-time-in-s l2 l3) 660)
        (= (drive-time-in-s l2 l4) 1380)
        (= (drive-time-in-s l3 l4) 1140)

        (= (drive-time-in-s l1 g1) 1080)
        (= (drive-time-in-s l2 g1) 1140)
        (= (drive-time-in-s l3 g1) 720)
        (= (drive-time-in-s l4 g1) 1140)
        (= (drive-time-in-s l2 l1) 1140)
        (= (drive-time-in-s l3 l1) 1200)
        (= (drive-time-in-s l4 l1) 1920)
        (= (drive-time-in-s l3 l2) 660)
        (= (drive-time-in-s l4 l2) 1380)
        (= (drive-time-in-s l4 l3) 1140)
    )

    (:goal (and (at p1 l4)
                (at p2 l2)
    ))

    (:metric minimize (total-cost))
)
```

7.3. Problem B

```
(define (problem first_problem)
    (:domain transportation_service)
    (:objects p1 p2 p3 p4 p5 - passenger
              t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11 t12 t13 t14 t15 t16 t17
t18 t19 t20 - taxi

              l0 l1 l2 l3 l4 l5 l6 l7 l8 l9 l10 l11 l12 l13 l14 l15 l16
l17 l18 l19 l20 l21
              l22 l23 l24 l25 g1 g2 - location)
    (:init
```


```
        ; Initialize the total cost of the plan
        (= (total-cost) 0)

        ; Initialize infrastructure
        (garage g1)
        (garage g2)

        ; Initialize vehicle positions
        (at t1 g1)
        (= (cost-per-second t1) 0.069)
        (= (cost-per-km t1) 3.5)
        (driver-available t1)
        (at t2 g1)
        (= (cost-per-second t2) 0.069)
        (= (cost-per-km t2) 3.5)
        (driver-available t2)
        (at t3 g1)
        (= (cost-per-second t3) 0.069)
        (= (cost-per-km t3) 3.5)
        (driver-available t3)
        (at t4 g1)
        (= (cost-per-second t4) 0.069)
        (= (cost-per-km t4) 3.5)
        (driver-available t4)
        (at t5 g1)
        (= (cost-per-second t5) 0.069)
        (= (cost-per-km t5) 3.5)
        (driver-available t5)
        (at t6 g1)
        (= (cost-per-second t6) 0.069)
        (= (cost-per-km t6) 3.5)
        (driver-available t6)
        (at t7 g1)
        (= (cost-per-second t7) 0.069)
        (= (cost-per-km t7) 3.5)
        (driver-available t7)
        (at t8 g1)
        (= (cost-per-second t8) 0.069)
        (= (cost-per-km t8) 3.5)
```


```
        (driver-available t8)
        (at t9 g1)
        (= (cost-per-second t9) 0.069)
        (= (cost-per-km t9) 3.5)
        (driver-available t9)
        (at t10 g1)
        (= (cost-per-second t10) 0.069)
        (= (cost-per-km t10) 3.5)
        (driver-available t10)
        (at t11 g1)
        (= (cost-per-second t11) 0.069)
        (= (cost-per-km t11) 3.5)
        (driver-available t11)
        (at t12 g1)
        (= (cost-per-second t12) 0.069)
        (= (cost-per-km t12) 3.5)
        (driver-available t12)
        (at t13 g1)
        (= (cost-per-second t13) 0.069)
        (= (cost-per-km t13) 3.5)
        (driver-available t13)
        (at t14 g1)
        (= (cost-per-second t14) 0.069)
        (= (cost-per-km t14) 3.5)
        (driver-available t14)
        (at t15 g1)
        (= (cost-per-second t15) 0.069)
        (= (cost-per-km t15) 3.5)
        (driver-available t15)
        (at t16 g1)
        (= (cost-per-second t16) 0.069)
        (= (cost-per-km t16) 3.5)
        (driver-available t16)
        (at t17 g1)
        (= (cost-per-second t17) 0.069)
        (= (cost-per-km t17) 3.5)
        (driver-available t17)
        (at t18 g1)
        (= (cost-per-second t18) 0.069)
```


```
        (= (cost-per-km t18) 3.5)
        (driver-available t18)
        (at t19 g1)
        (= (cost-per-second t19) 0.069)
        (= (cost-per-km t19) 3.5)
        (driver-available t19)
        (at t20 g1)
        (= (cost-per-second t20) 0.069)
        (= (cost-per-km t20) 3.5)
        (driver-available t20)


        ; Initilize vehicle properties
        (cost-calc-mutex t1)
        (cost-calc-mutex t2)
        (cost-calc-mutex t3)
        (cost-calc-mutex t4)
        (cost-calc-mutex t5)
        (cost-calc-mutex t6)
        (cost-calc-mutex t7)
        (cost-calc-mutex t8)
        (cost-calc-mutex t9)
        (cost-calc-mutex t10)
        (cost-calc-mutex t11)
        (cost-calc-mutex t12)
        (cost-calc-mutex t13)
        (cost-calc-mutex t14)
        (cost-calc-mutex t15)
        (cost-calc-mutex t16)
        (cost-calc-mutex t17)
        (cost-calc-mutex t18)
        (cost-calc-mutex t19)
        (cost-calc-mutex t20)

        (start-vehicle-cost t1)
        (start-vehicle-cost t2)
        (start-vehicle-cost t3)
        (start-vehicle-cost t4)
        (start-vehicle-cost t5)
```


```
        (start-vehicle-cost t6)
        (start-vehicle-cost t7)
        (start-vehicle-cost t8)
        (start-vehicle-cost t9)
        (start-vehicle-cost t10)
        (start-vehicle-cost t11)
        (start-vehicle-cost t12)
        (start-vehicle-cost t13)
        (start-vehicle-cost t14)
        (start-vehicle-cost t15)
        (start-vehicle-cost t16)
        (start-vehicle-cost t17)
        (start-vehicle-cost t18)
        (start-vehicle-cost t19)
        (start-vehicle-cost t20)

        (leave-passenger-mutex t1)
        (leave-passenger-mutex t2)
        (leave-passenger-mutex t3)
        (leave-passenger-mutex t4)
        (leave-passenger-mutex t5)
        (leave-passenger-mutex t6)
        (leave-passenger-mutex t7)
        (leave-passenger-mutex t8)
        (leave-passenger-mutex t9)
        (leave-passenger-mutex t10)
        (leave-passenger-mutex t11)
        (leave-passenger-mutex t12)
        (leave-passenger-mutex t13)
        (leave-passenger-mutex t14)
        (leave-passenger-mutex t15)
        (leave-passenger-mutex t16)
        (leave-passenger-mutex t17)
        (leave-passenger-mutex t18)
        (leave-passenger-mutex t19)
        (leave-passenger-mutex t20)

        ; List of booked passangers
        (at p1 l13)
```


```
        (= (handling-time p1) 120)
        (at 45000 (ready-for-pickup p1))
        (at 48600 (not (ready-for-pickup p1)))

        (at p2 l0)
        (= (handling-time p2) 120)
        (at 44100 (ready-for-pickup p2))
        (at 47700 (not (ready-for-pickup p2)))

        (at p3 l1)
        (= (handling-time p3) 120)
        (at 44100 (ready-for-pickup p3))
        (at 47700 (not (ready-for-pickup p3)))

        (at p4 l4)
        (= (handling-time p4) 180)
        (at 44100 (ready-for-pickup p4))
        (at 47700 (not (ready-for-pickup p4)))
; Connect all roads and set their driving distance/estimated time
values
        (road l1 l10)
        (= (drive-time-in-s l1 l10) 23162.5)
        (= (drive-distance-in-km l1 l10) 339.146)
        (road l1 l4)
        (= (drive-time-in-s l1 l4) 282.4)
        (= (drive-distance-in-km l1 l4) 1.839)
        (road l1 l22)
        (= (drive-time-in-s l1 l22) 335.6)
        (= (drive-distance-in-km l1 l22) 3.664)
        (road l1 l9)
        (= (drive-time-in-s l1 l9) 463.8)
        (= (drive-distance-in-km l1 l9) 5.37)
        (road l1 l18)
        (= (drive-time-in-s l1 l18) 396.4)
        (= (drive-distance-in-km l1 l18) 4.188)
        (road l1 l25)
        (= (drive-time-in-s l1 l25) 496.4)
        (= (drive-distance-in-km l1 l25) 3.98)
        (road l1 l8)
```


```
        (= (drive-time-in-s l1 l8) 437.9)
        (= (drive-distance-in-km l1 l8) 3.628)
        (road l1 l17)
        (= (drive-time-in-s l1 l17) 352.3)
        (= (drive-distance-in-km l1 l17) 3.49)
        (road l1 l23)
        (= (drive-time-in-s l1 l23) 744.2)
        (= (drive-distance-in-km l1 l23) 8.63)
        (road l1 l20)
        (= (drive-time-in-s l1 l20) 296.4)
        (= (drive-distance-in-km l1 l20) 2.294)
        (road l1 l6)
        (= (drive-time-in-s l1 l6) 535.4)
        (= (drive-distance-in-km l1 l6) 4.298)
        (road l1 l0)
        (= (drive-time-in-s l1 l0) 590.2)
        (= (drive-distance-in-km l1 l0) 3.567)
        (road l1 l15)
        (= (drive-time-in-s l1 l15) 309.2)
        (= (drive-distance-in-km l1 l15) 2.388)
        (road l1 g2)
        (= (drive-time-in-s l1 g2) 591)
        (= (drive-distance-in-km l1 g2) 6.993)
        (road l1 l13)
        (= (drive-time-in-s l1 l13) 333.2)
        (= (drive-distance-in-km l1 l13) 2.949)
        (road l1 l7)
        (= (drive-time-in-s l1 l7) 670.6)
        (= (drive-distance-in-km l1 l7) 5.427)
        (road l1 l5)
        (= (drive-time-in-s l1 l5) 537.7)
        (= (drive-distance-in-km l1 l5) 4.439)
        (road l1 l19)
        (= (drive-time-in-s l1 l19) 447.2)
        (= (drive-distance-in-km l1 l19) 3.695)
        (road l1 l2)
        (= (drive-time-in-s l1 l2) 319.7)
        (= (drive-distance-in-km l1 l2) 2.154)
        (road l1 l16)
```


```
        (= (drive-time-in-s l1 l16) 525.4)
        (= (drive-distance-in-km l1 l16) 3.936)
        (road l1 l3)
        (= (drive-time-in-s l1 l3) 441.4)
        (= (drive-distance-in-km l1 l3) 3.632)
        (road l1 l11)
        (= (drive-time-in-s l1 l11) 502.5)
        (= (drive-distance-in-km l1 l11) 4.068)
        (road l1 l14)
        (= (drive-time-in-s l1 l14) 358.4)
        (= (drive-distance-in-km l1 l14) 3.066)
        (road l1 g1)
        (= (drive-time-in-s l1 g1) 532.5)
        (= (drive-distance-in-km l1 g1) 6.436)
        (road l1 l21)
        (= (drive-time-in-s l1 l21) 1156.8)
        (= (drive-distance-in-km l1 l21) 13.87)
        (road l10 l1)
        (= (drive-time-in-s l10 l1) 23198.2)
        (= (drive-distance-in-km l10 l1) 339.232)
        (road l10 l4)
        (= (drive-time-in-s l10 l4) 23175.9)
        (= (drive-distance-in-km l10 l4) 339.269)
        (road l10 l22)
        (= (drive-time-in-s l10 l22) 23132.6)
        (= (drive-distance-in-km l10 l22) 337.609)
        (road l10 l9)
        (= (drive-time-in-s l10 l9) 23152)
        (= (drive-distance-in-km l10 l9) 337.891)
        (road l10 l18)
        (= (drive-time-in-s l10 l18) 23193.4)
        (= (drive-distance-in-km l10 l18) 338.133)
        (road l10 l25)
        (= (drive-time-in-s l10 l25) 23358.6)
        (= (drive-distance-in-km l10 l25) 340.662)
        (road l10 l8)
        (= (drive-time-in-s l10 l8) 23321.7)
        (= (drive-distance-in-km l10 l8) 340.728)
        (road l10 l17)
```


```
        (= (drive-time-in-s l10 l17) 23273.9)
        (= (drive-distance-in-km l10 l17) 339.339)
        (road l10 l23)
        (= (drive-time-in-s l10 l23) 23554.3)
        (= (drive-distance-in-km l10 l23) 344.981)
        (road l10 l20)
        (= (drive-time-in-s l10 l20) 23106.5)
        (= (drive-distance-in-km l10 l20) 338.645)
        (road l10 l6)
        (= (drive-time-in-s l10 l6) 23428.9)
        (= (drive-distance-in-km l10 l6) 341.729)
        (road l10 l0)
        (= (drive-time-in-s l10 l0) 23474)
        (= (drive-distance-in-km l10 l0) 340.667)
        (road l10 l15)
        (= (drive-time-in-s l10 l15) 23119.3)
        (= (drive-distance-in-km l10 l15) 338.739)
        (road l10 g2)
        (= (drive-time-in-s l10 g2) 23080.9)
        (= (drive-distance-in-km l10 g2) 336.969)
        (road l10 l13)
        (= (drive-time-in-s l10 l13) 23217)
        (= (drive-distance-in-km l10 l13) 340.049)
        (road l10 l7)
        (= (drive-time-in-s l10 l7) 23554.4)
        (= (drive-distance-in-km l10 l7) 342.527)
        (road l10 l5)
        (= (drive-time-in-s l10 l5) 23370.7)
        (= (drive-distance-in-km l10 l5) 340.782)
        (road l10 l19)
        (= (drive-time-in-s l10 l19) 23319.7)
        (= (drive-distance-in-km l10 l19) 340.274)
        (road l10 l2)
        (= (drive-time-in-s l10 l2) 23213.2)
        (= (drive-distance-in-km l10 l2) 339.584)
        (road l10 l16)
        (= (drive-time-in-s l10 l16) 23418.9)
        (= (drive-distance-in-km l10 l16) 341.366)
        (road l10 l3)
```


```
        (= (drive-time-in-s l10 l3) 23325.2)
        (= (drive-distance-in-km l10 l3) 340.732)
        (road l10 l11)
        (= (drive-time-in-s l10 l11) 23396)
        (= (drive-distance-in-km l10 l11) 341.498)
        (road l10 l14)
        (= (drive-time-in-s l10 l14) 23242.2)
        (= (drive-distance-in-km l10 l14) 340.166)
        (road l10 g1)
        (= (drive-time-in-s l10 g1) 23342.6)
        (= (drive-distance-in-km l10 g1) 342.787)
        (road l10 l21)
        (= (drive-time-in-s l10 l21) 23156.8)
        (= (drive-distance-in-km l10 l21) 337.004)
        (road l4 l1)
        (= (drive-time-in-s l4 l1) 287.5)
        (= (drive-distance-in-km l4 l1) 1.866)
        (road l4 l10)
        (= (drive-time-in-s l4 l10) 23157.5)
        (= (drive-distance-in-km l4 l10) 339.217)
        (road l4 l22)
        (= (drive-time-in-s l4 l22) 437.8)
        (= (drive-distance-in-km l4 l22) 4.246)
        (road l4 l9)
        (= (drive-time-in-s l4 l9) 373.1)
        (= (drive-distance-in-km l4 l9) 3.477)
        (road l4 l18)
        (= (drive-time-in-s l4 l18) 498.6)
        (= (drive-distance-in-km l4 l18) 4.77)
        (road l4 l25)
        (= (drive-time-in-s l4 l25) 448.6)
        (= (drive-distance-in-km l4 l25) 3.355)
        (road l4 l8)
        (= (drive-time-in-s l4 l8) 390.1)
        (= (drive-distance-in-km l4 l8) 3.004)
        (road l4 l17)
        (= (drive-time-in-s l4 l17) 279.9)
        (= (drive-distance-in-km l4 l17) 2.203)
        (road l4 l23)
```


```
        (= (drive-time-in-s l4 l23) 739.2)
        (= (drive-distance-in-km l4 l23) 8.701)
        (road l4 l20)
        (= (drive-time-in-s l4 l20) 291.4)
        (= (drive-distance-in-km l4 l20) 2.365)
        (road l4 l6)
        (= (drive-time-in-s l4 l6) 291.9)
        (= (drive-distance-in-km l4 l6) 2.597)
        (road l4 l0)
        (= (drive-time-in-s l4 l0) 542.4)
        (= (drive-distance-in-km l4 l0) 2.942)
        (road l4 l15)
        (= (drive-time-in-s l4 l15) 304.2)
        (= (drive-distance-in-km l4 l15) 2.459)
        (road l4 g2)
        (= (drive-time-in-s l4 g2) 586)
        (= (drive-distance-in-km l4 g2) 7.064)
        (road l4 l13)
        (= (drive-time-in-s l4 l13) 285.4)
        (= (drive-distance-in-km l4 l13) 2.325)
        (road l4 l7)
        (= (drive-time-in-s l4 l7) 575.5)
        (= (drive-distance-in-km l4 l7) 5.416)
        (road l4 l5)
        (= (drive-time-in-s l4 l5) 489.9)
        (= (drive-distance-in-km l4 l5) 3.815)
        (road l4 l19)
        (= (drive-time-in-s l4 l19) 399.4)
        (= (drive-distance-in-km l4 l19) 3.07)
        (road l4 l2)
        (= (drive-time-in-s l4 l2) 129.2)
        (= (drive-distance-in-km l4 l2) 0.898)
        (road l4 l16)
        (= (drive-time-in-s l4 l16) 281.9)
        (= (drive-distance-in-km l4 l16) 2.235)
        (road l4 l3)
        (= (drive-time-in-s l4 l3) 370.9)
        (= (drive-distance-in-km l4 l3) 3.876)
        (road l4 l11)
```


```
        (= (drive-time-in-s l4 l11) 259)
        (= (drive-distance-in-km l4 l11) 2.367)
        (road l4 l14)
```

...

```
    )

    (:goal (and (at p1 l1)
                (at p2 l22)
                (at p3 l10)
                (at p4 l10)
    ))



    (:metric minimize (total-cost))
)
```