Seminar Unit 6

Path, route and distribution planning as optimisation problems

Optimización de problemas de distribución y rutas

Contents Part 1

- Distributions problems and freight load/unload services
- Vehicle and path problems
- Route and timetable optimisation

The general **goal** is to **find the set of routes** (plus its **temporal + resource plan**) **of lowest cost**, selecting the most adequate vehicles that satisfy the freight characteristics and constraints

But other **performance indicators** are also valid

Indicator	Description	Formula
Deadline	Time for the customer to receive the goods	Plazo de entrega = $= \frac{\sum_{pedidos} (\text{Fecha de aceptación} - \text{Fecha de solicitud})}{\text{Total pedidos entregados}}$
Adequate to the deadline	Useful to monitor urgent orders or with fixed deadlines	Fecha de entrega = $= \frac{\text{Número de pedidos (*) (líneas) en fecha}}{\text{Total pedidos recibidos}} \times 100$
Stock days (*rotation index*)	Usually defined yearly for each product	$\text{Rotación} = \frac{\sum \text{Salidas}}{\text{Cantidad media de stock}}$
Obsolescence index	Yearly defined for each product	$\text{Obsolencia} = \frac{\text{Entregas al año}}{\text{Cantidad media de stock}}$
Breakdown index	Defined for a given period and for each product	$\text{Rotura} = \frac{\text{Pedidos no satisfechos}}{\text{Pedidos totales}} \times 100$

General aspects to be considered:

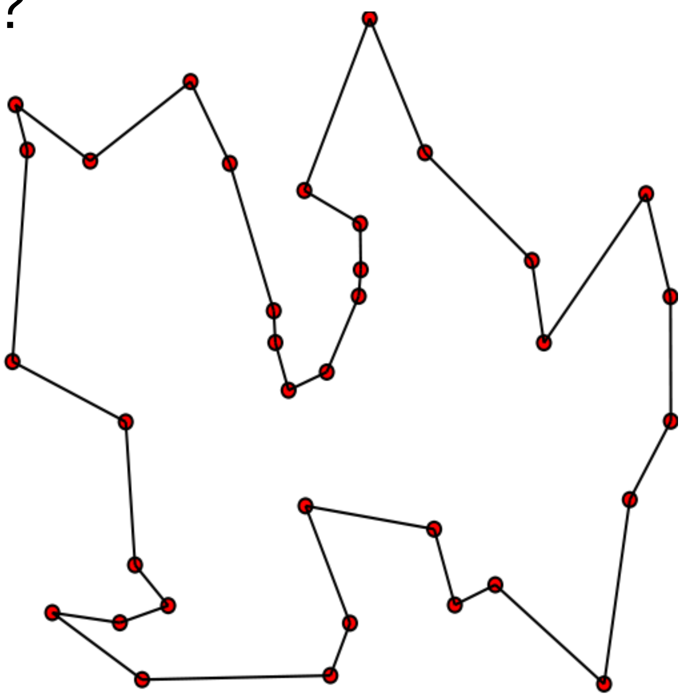
- Each order means a type of load (weight, volume, fragility, etc.), urgency, origin, destination, service time (operating time), deadline, repetition period, etc.
  - The type of load modifies the times and operating costs
- A static and permanent shipping network is common, with transfer locations (with different capacities, working shifts and particular constraints)
  - The number of paths can be huge, so it is usually limited a priori – the complexity can be drastically reduced

General aspects to be considered:

- Hiring costs and vehicle usage, including:
  - Fixed/variable, direct/indirect costs, especially if they have to be outsourced; and availability, as they are not always where they are needed
  - Stopping (waiting) cost of vehicles in transfer locations (and for drivers)
  - Satisfying all capacity constraints
- Don't forget that different customers can be charged with different fees (customised policies) and customers require/use different timetables

TSP (Travelling Salesman Person) as a **special case** of the **TPP** (Travelling Purchaser Problem)

Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits **each city** exactly once and returns to the origin city?



Model as a **weighted graph**

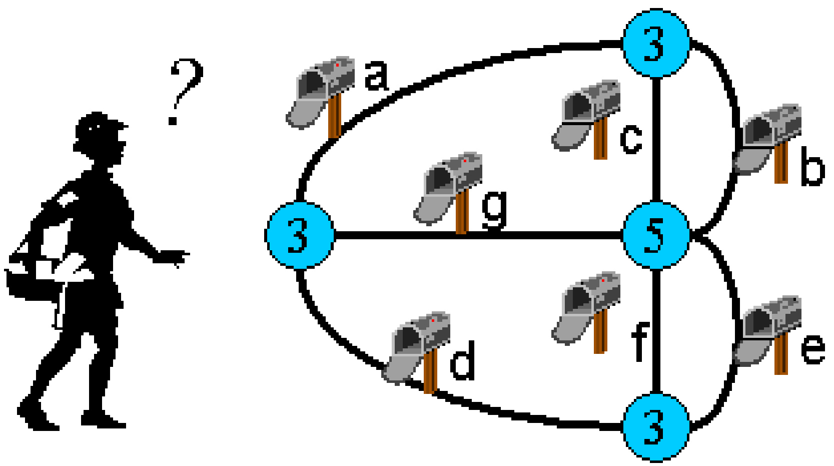
- nodes: cities
- edges: (un)directed paths between two cities
- paths are labelled (distance, cost, resource use, etc.)
- optimise the Hamiltonian path

CPP (Chinese Postman Problem)

Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits **each street** exactly once and returns to the origin city?

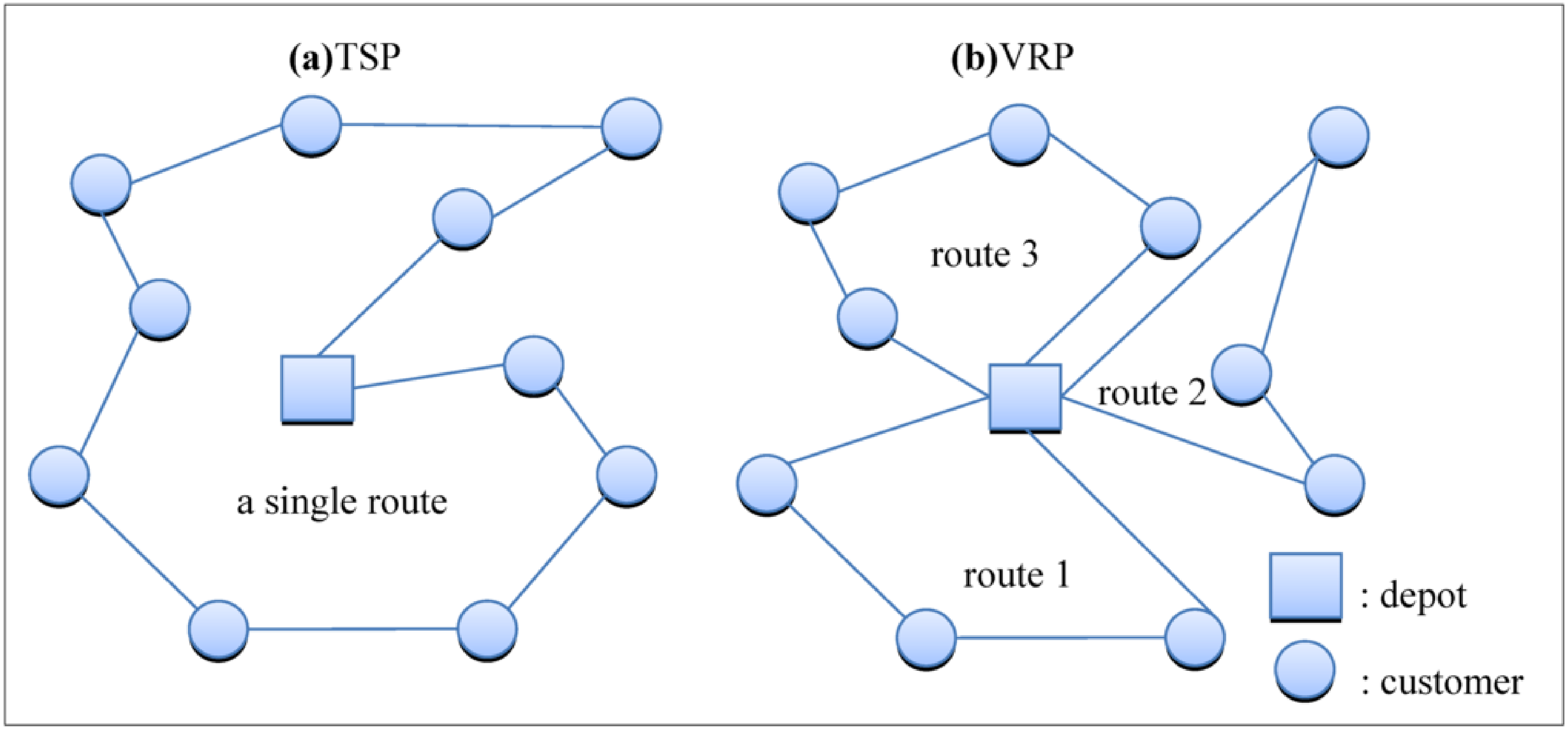
Same model as before

- now, the **optimal solution** is to visit **each edge only once** (Eulerian path)
- the **start** and **end node** are **the same** (Eulerian cycle)



VRP (Vehicle Routing Problem) as a generalisation of the TPP

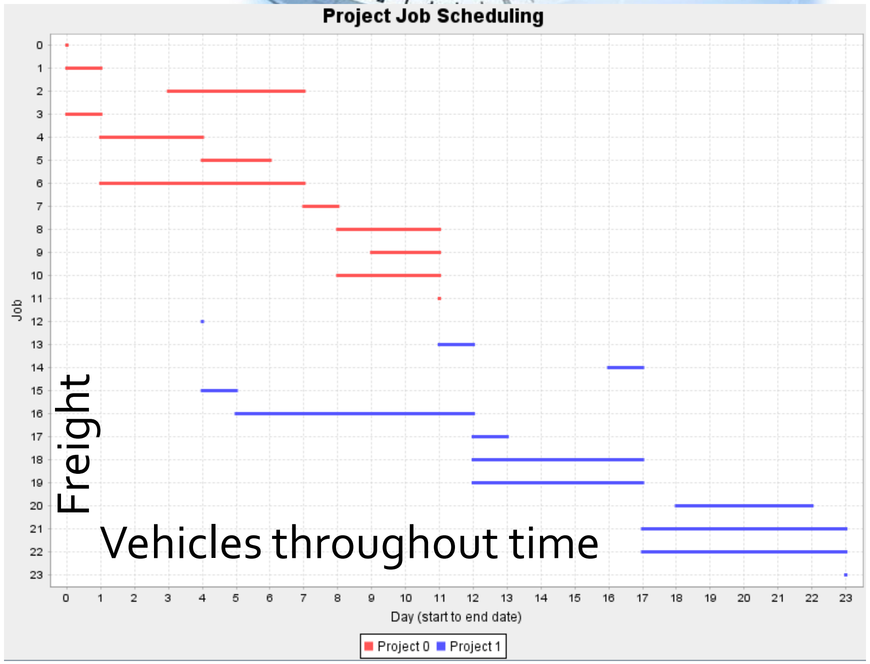
What is the optimal set of routes for a fleet of vehicles to traverse to deliver to a given set of customers?



Scheduling for **resource assignment (timetable)**

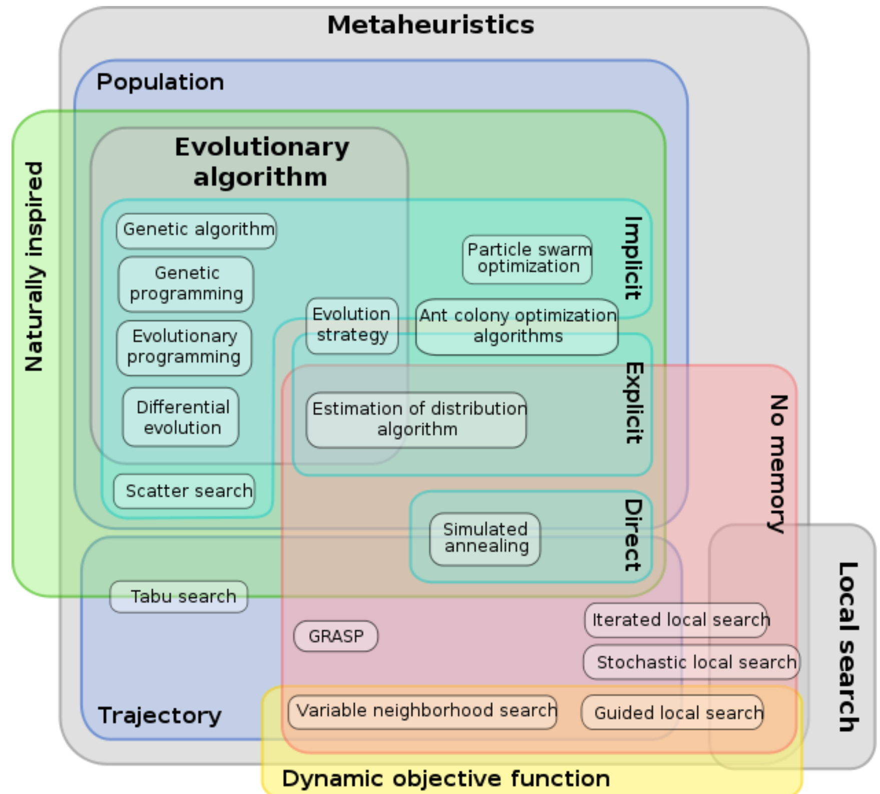
Model

- **tasks** to be done
- finite **capacity resources** to be used
- **mapping** of tasks to resources and processing **times**, **subject** to feasibility **constraints** (due times, precedence relations, priorities, costs, etc.) and **optimisation** objectives



How to solve (optimise) them?

- Search: blind search, **heuristic search**, A algorithms, A*, IDA, IDA*, etc.
- Constraint programming
- Metaheuristics:
  - Local search
  - Genetic algorithms
  - Ant colony optimisation
  - Simulated annealing
- Linear programming



Task 1. Assignment of demands to vehicles

Freight, as a number of *n* demands: {d1, d2… dn}

- weight {w1, w2… wn} (and probably volume {v1, v2… vn})
- origin, destination {<o1,d1>, <o2,d2>… <on,dn>}

Vehicles, as a number of *m* vehicles: {v1, v2… vm} (we assume each one has its own driver)

- max weight/volume {vw1, vw2… vwm} and {vv1, vv2… vvm}
- cost {vc1, vc2… vcm}

Goal: minimise the cost

Task 2. Task 1 + selection of routes

Freight, as a number of *n* demands: {d1, d2… dn}

due time {t1, t2… tn}

Vehicles, as a number of *m* vehicles: {v1, v2… vm}

each vehicle can use different **routes**

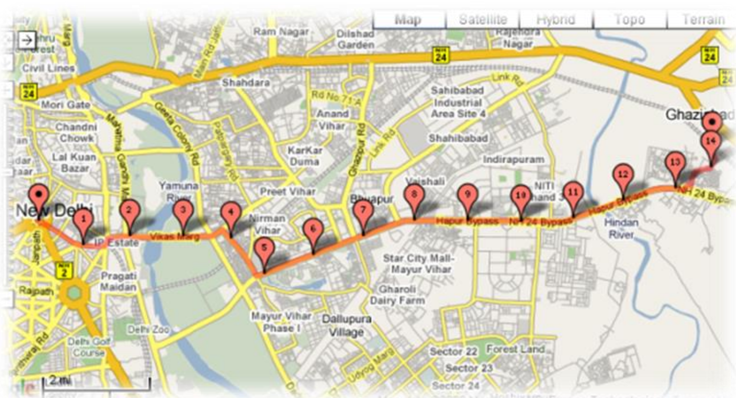
eg. v1: {r1, r2… ri}; v2: {r1, r3… rj}; vm: {r2, r4}

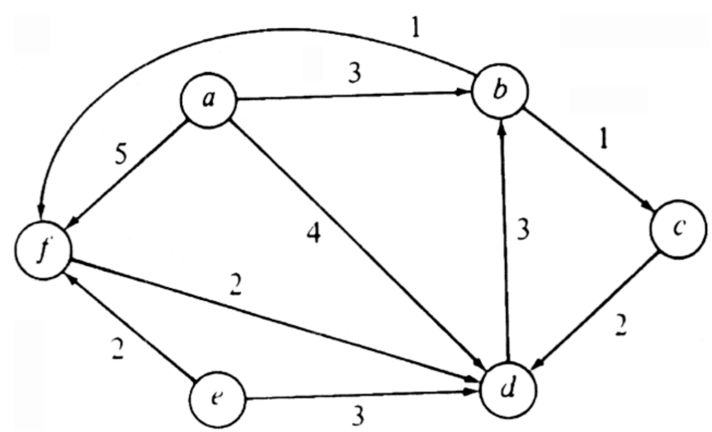
each route has a cost {cr1, cr2… cri… crj} and duration {dr1, dr2… dri… drj}

Task 3. Task 2 + creation and selection of routes

Route, which needs to be generated dynamically in terms of a sequence of cities (nodes)

- each route will have a **cost** and **duration** that **depends on the arcs** between cities

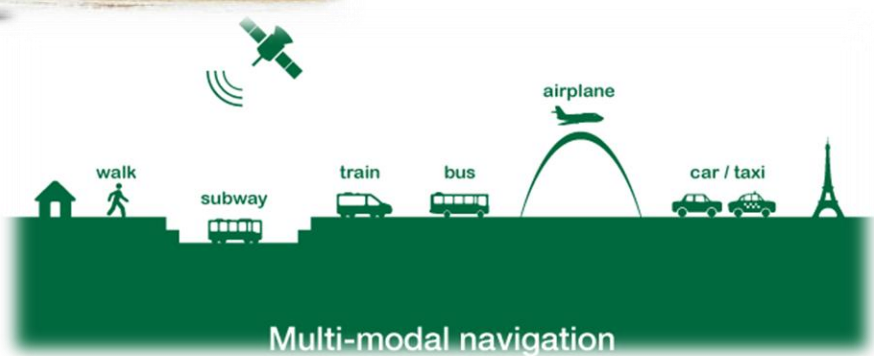




Contents Part 2

- Multimodal and intermodal transport
- New advances: synchromodal transport

The general **goal** is to **find the set of routes**, with everything that they involve (temporal + resource constraints), **selecting the most adequate transport types (multimodality)**



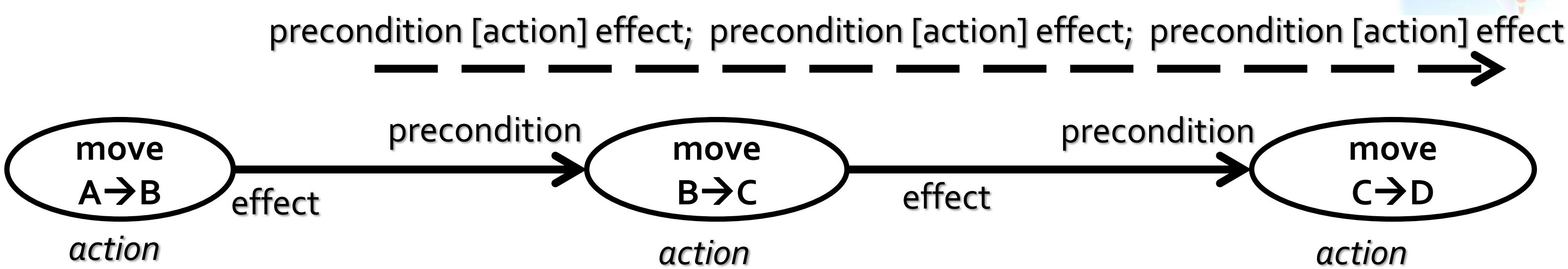
How to solve (optimise) them?

- Search: blind search, **heuristic search**, A algorithms, A*, IDA, IDA*, etc.
- Constraint programming
- Metaheuristics: local search, genetic algorithms, etc.
- **Planning technology**
  - PDDL (Planning Domain Definition Language)
  - Modelling actions (i.e. rules) with preconditions and effects
  - Goal: **plan**, which allows us to get the goals

Planning - Definition

Planning is an **intelligent** reasoning **process** to choose and organise, that is to build a plan as a collection of actions that allow us to reach a set of goals starting from an initial state

Causal links



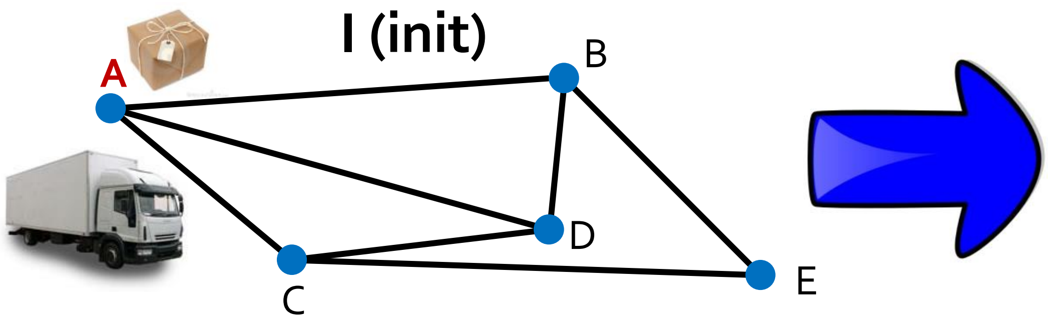
More formally, a **planning problem** is defined by the 3-tuple **<I,G,A>**:

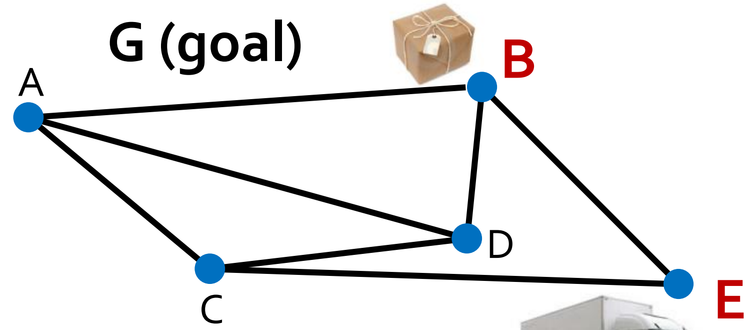
- **I**: initial state, with a complete assignment of variables
- **G**: goals to be achieved, with a partial assignment of variables
- **A**: actions, in the form <pre, add, del> that modify the current state and define the sequence of caulsal links

Plan: collection of actions {a1,a2… an} that transforms I into G

- the **ordering** between actions is **important**, as one action a1 can remove something that a2 needs

Example





V truck∈ {A,B,C,D,E}
parcel∈ {A,B,C,D,E,truck}

I truck: A
parcel: A

G truck: E
parcel: B

A **move**(?vehicle, ?locORIG, ? locDEST)
pre: ?vehicle in ?locORIG
add: ?vehicle in ?locDEST
del: ?vehicle in ? locORIG
load(?parcel, ?vehicle, ? location)
pre: ?parcel in ?location
?vehicle in ?location
add: ?parcel in ?vehicle
del: ?parcel in ?location
unload(?parcel, ? vehicle, ?location)
...

PDDL (Planning Domain Definition Language) consists of **two** plain text **files** *http://cs-www.cs.yale.edu/homes/dvm*

- **Domain**, defines the predicates, functions and actions that can be planned, no matter the particular problem
- **Problem**, defines the initial state and the goals in terms of the predicates of the domain; it also includes the metric to be optimised
- 1 domain associated to many problems; 1 problem only to 1 domain

```
(define (domain <domain name>)
   <PDDL code for predicates>
   <PDDL code for first action>
   [...]
   <PDDL code for last action>
)
```

```
(define (problem <problem name>)
   (:domain <domain name>)
   <PDDL code for objects>
   <PDDL code for initial state>
   <PDDL code for goal specification>
)
```

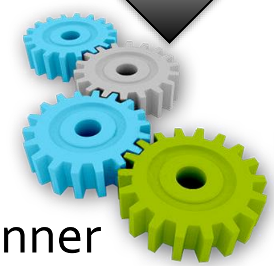

A PDDL logistics example

What happens with durations?

```
(define (domain logistics)
(:requirements :strips :typing)
(:types
        package location vehicle - object
        truck - vehicle
        city - location)
(:predicates
        (at ?vehicle-or-package - (either vehicle package)  ?location - location)
        (in ?package - package ?vehicle - vehicle))
(:action load  ;action to load a package in a truck
  :parameters (?obj - package
               ?truck - truck
               ?loc - location)
  :precondition (and   (at ?truck ?loc)
                       (at ?obj ?loc))
  :effect (and (not (at ?obj ?loc))
               (in ?obj ?truck)))
(:action unload ; action to unload a package from a truck
  :parameters  (?obj – package
                ?truck - truck
                ?loc - location)
  :precondition (and (at ?truck ?loc)
                     (in ?obj ?truck))
  :effect (and   (not (in ?obj ?truck))
                 (at ?obj ?loc)))
(:action move ... ;action to move between two locations
```

```
(define (problem pb1)
(:domain logistics)
(:requirements :strips :typing)
(:objects  pck1- package
           pck2 - package
           pck3 – package
           truck1 – vehicle
           truck2 – vehicle
           madrid – city
           valencia – city
           barcelona – city ...)
(:init     (at truck1 valencia) (at truck2 madrid)
           (at pck1 barcelona) (at pck2 madrid)...)
(:goals (and (at pck1 madrid)
             (at pck2 valencia)
             (at truck1 madrid)
             (at truck2 barcelona)))
```


PDDL encoding



Planner

(define (domain logistics)
(:requirements :strips :typing)
(:types
    package location vehicle - object
    truck - vehicle
    city - location)
(:predicates
    (at ?vehicle-or-package - (either vehicle package) …

FF, LPG,
OPTIC, etc.

0: (move truck4 city3 city4)
0: (walk driver1 city1 …
70: (move truck3 city2 …
134:(move truck2 city1 …
134:(load package5 truck3 …

PDDL domain in more detail

(define (domain <name>)

(:requirements <:req 1>... <:req n>) *; planning requirements*

(:types <subtype1>... <subtype n> – <type1> <typen>) *; types & subtypes to be used*

(:constants <cons1> ... <consn>) *; constants to be used*

(:predicates <p1> <p2>... <pn>) *; predicates (true/false info) – for objets*

(:functions <f1> <f2>... <fn>) *; functions (fluents or numeric info) – for resources*

PDDL domain in more detail

```
; example of one action
(:durative-action act1        ; action with duration
    :parameters (?par1 – <subtype1> ?par2 – <subtype2> ...) ; the needed parameters
    :duration <value>    ; duration of the action
    :condition (and (at start (<condition_1>))  ; "at start", "over all", "at end"
                    (over all (<condition_2>))
                    ...
                    (at end (<condition_n>)))
    :effect (and    (at start (<effect_1>))        ; "at start", "at end"
                    (at end (<effect_2>))
                    ...
                    (at end (not (<effect_n>))))
)
```


PDDL problem in more detail

(define (problem <name>)

(:domain <name >) *; domain this problem belongs to*

(:objects <obj$_1$> - <type$_1$> … <obj$_n$> - <type$_n$>) *; objects and their types*

(:init *; initial state*

(<predicate$_1$>) … (<predicate$_i$>) *; true/false predicates (propositional)*

(= <function$_1$> <value$_1$>) … (= <function$_n$> <value$_n$>)) *; numeric info*

(:goal *; goals*

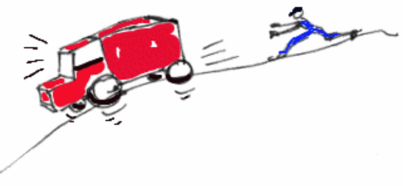
(and ((<predicate$_1$>) … (<predicate$_i$>) *; propositional goals*

(<operator$_1$> <function$_1$> <value$_1$>) …

(<operator$_j$> <function$_j$> <value$_j$>))) *; numeric goals*

(:metric minimize|maximize <expression>**))** *; metric to min/max – plan quality*

Driverlog scenario (mono-modal) as a basis



This domain has **drivers** that can **walk** between **locations** and **trucks** that can **drive** between locations.

Walking requires traversal of **different paths from** those **used for driving**, and there is always one intermediate location on a footpath between two road junctions.

The **trucks** can be **loaded or unloaded** with **packages** (with or without a **driver** present) and the **objective** is to **transport packages** between locations, ending up with a subset of the packages, the trucks and the drivers **at specified destinations**.

The **metric** adds **costs** for walking and driving and problem instances required that the planner **optimise** some **linear combination** of them

Task 1. Generation of a multi-modal plan in PDDL

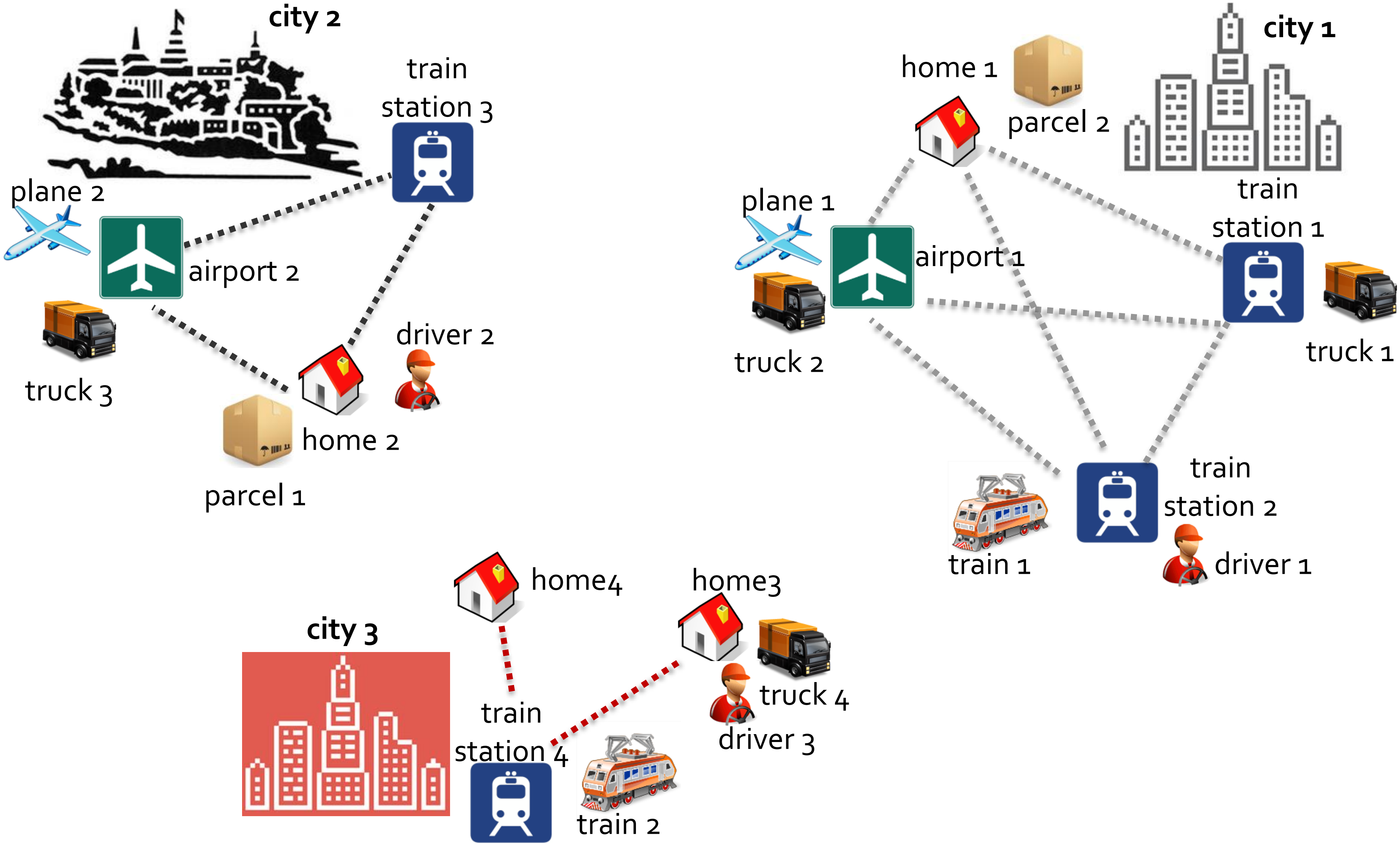
City, as the area where the locations are placed

Location, as the origin/destination of parcels

Parcel/freight, as the items to be transported by vehicles

Vehicles, particularly trucks (with **drivers**), trains & planes

Goal: minimise the makespan (plan duration)


city 2
train station 3
plane 2
airport 2
truck 3
driver 2
home 2
parcel 1
home 1
parcel 2
city 1
plane 1
airport 1
train station 1
truck 2
truck 1
train station 2
train 1
driver 1
home4
home3
city 3
truck 4
driver 3
train station 4
train 2

Some additional constraints to model in the actions

- Trucks can only move between locations of the same city; trucks need a driver that can also walk between locations of the same city
- Trucks have capacity constraints on the weight/volume or type of parcel to be transported
- Trains/planes only move between train stations/airports (no matter the city)

- Problem **goal**: transport parcel1 and parcel2 to home4 in city3

Some additional constraints to model in the actions

- **Define the PDDL domain+problem files (with durative actions and metric)**

- Additional (valuable) **extensions:**
  - Paths between locations within a city
  - Numeric cost & capacities in trains/planes
  - Use of additional resources, e.g. pilot, crane to (un)load, extra staff, etc.
  - Different types of parcels and delivery deadlines, etc.

Contents Part 3

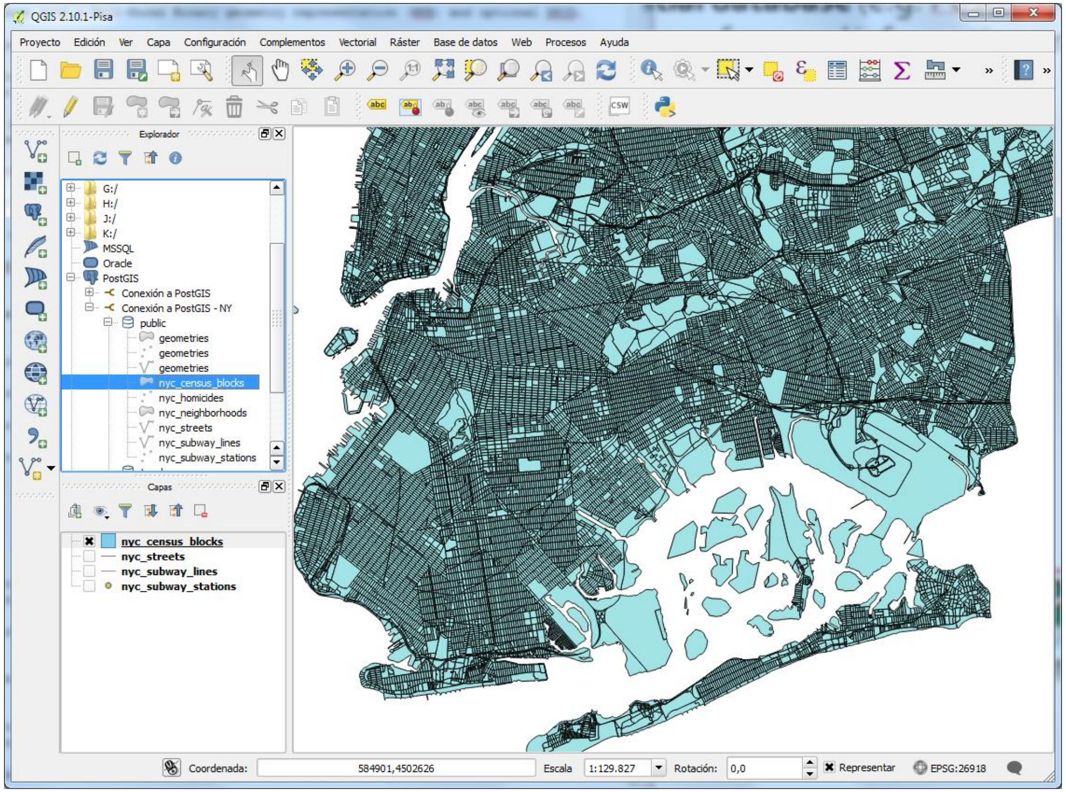
- Geographical Information Systems (GIS) for transportation (GIS-T) and logistics
- GIS for decision taking

The general **goal** is to use a **spatial database** (e.g. PostGIS) to work with SQL extensions for georeferenced information and combine that with a GIS tool (e.g. QGIS)

```
SELECT
  id, name,
  ST_Distance(geog,
              'POINT(-73.98 40.77)')
FROM geonames
WHERE ts @@
      to_tsquery('english','oak & tree')
AND ST_DWithin(geog,
      'POINT(-73.98 40.77)',
      100000);
```

```
  id    |                name                |  st_distance
--------+------------------------------------+----------------
 5102106 | Oak Tree                           | 39739.186642099
 5102107 | Oak Tree Grade School (historical) | 39658.142606346

Time: 5.337 ms
```

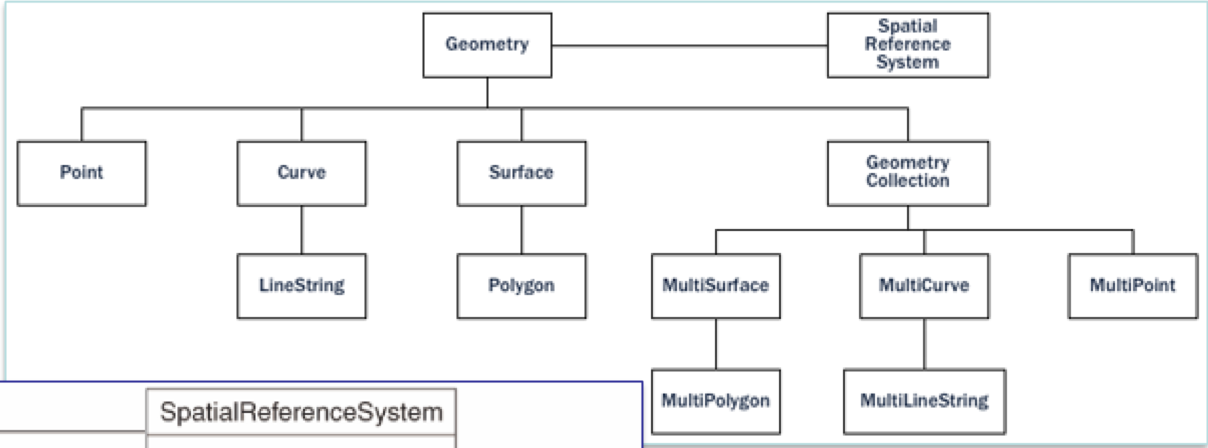


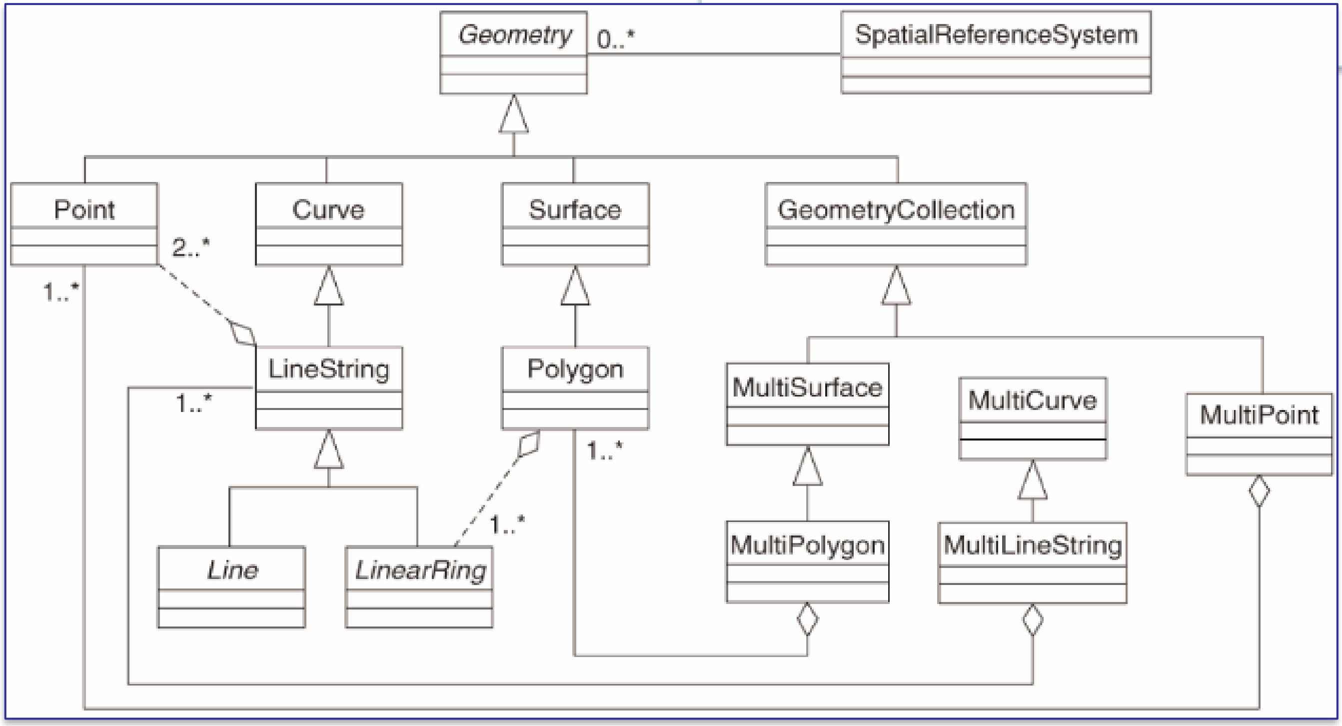
Spatial databases **store** and **manipulate spatial objects** like any other object in the database

Main **aspects:**

- Spatial data types refer to shapes such as point, line, and polygon
- Multi-dimensional spatial indexing is used for efficient processing of spatial operations
- Spatial functions in SQL for querying of spatial properties, analyzing geometric components, determining spatial relationships, and manipulating geometries

Geometry hierarchy





Model to to store geospatial data

Seminar.
Some useful functions

ST_GeometryType(geometry)	returns the type of the geometry
ST_NDims(geometry)	returns the number of dimensions of the geometry
ST_SRID(geometry)	returns the spatial reference identifier number of the geometry
ST_X(point)	returns the X coordinate
ST_Y(point)	returns the Y coordinate
ST_Length(linestring)	returns the length of the linestring
ST_StartPoint(geometry)	returns the first coordinate as a point
ST_EndPoint(geometry)	returns the last coordinate as a point
ST_NPoints(geometry)	returns the number of coordinates in the linestring
ST_Area(geometry)	returns the area of the polygons
ST_NRings(geometry)	returns the number of rings (usually 1, more if there are holes)
ST_ExteriorRing(polygon)	returns the outer ring as a linestring
ST_InteriorRingN(polygon, integer)	returns a specified interior ring as a linestring
ST_Perimeter(geometry)	returns the length of all the rings

Seminar. Some useful functions

ST_NumGeometries(multi/geomcollection)	returns the number of parts in the collection
ST_GeometryN(geometry, integer)	returns the specified part of the collection
ST_GeomFromText(text)	returns geometry
ST_AsText(geometry)	returns WKT text
ST_AsEWKT(geometry)	returns EWKT text
ST_GeomFromWKB(bytea)	returns geometry
ST_AsBinary(geometry)	returns WKB bytes
ST_AsEWKB(geometry)	returns EWKB bytes
ST_GeomFromGML(text)	returns geometry
ST_AsGML(geometry)	returns GML text
ST_GeomFromKML(text)	returns geometry
ST_AsKML(geometry)	returns KML text
ST_AsGeoJSON(geometry)	returns JSON text
ST_AsSVG(geometry)	returns SVG text

Seminar. Some examples

Fragments of A-3

```
SELECT *
FROM public.roads
WHERE REF = 'A-3';
```

Data Output | Explain | Messages | History

	gid integer	osm_id character varying(11)	name character varying(48)	ref character varying	type character varying(16)	oneway smallint	bridge smallint	tunnel smallint	maxspeed smallint	geom geometry(MultiLineSt
41	1695255	238843112	Autovía del Este	A-3	motorway	1	0	0		010500000001000000
42	1695256	238843113	Autovía del Este	A-3	motorway	1	0	0	120	010500000001000000
43	1695262	238843119	Autovía del Este	A-3	motorway	1	0	0	100	010500000001000000
44	1695264	238843121	Autovía del Este	A-3	motorway	1	0	0	100	010500000001000000
45	1695267	238843124	Autovía del Este	A-3	motorway	1	0	0	120	010500000001000000
46	1695270	238843127	Autovía del Este	A-3	motorway	1	0	0	120	010500000001000000
47	1695274	238843131		A-3	motorway	1	0	0		010500000001000000
48	1695301	238844863	Autovía del Este	A-3	motorway	1	0	0		010500000001000000
49	1695307	238844872	Autovía del Este	A-3	motorway	1	0	0		010500000001000000
50	1695310	238844875	Autovía del Este	A-3	motorway	1	0	0	120	010500000001000000

Total length of roads

```
SELECT Sum(ST_Length(geom))
FROM public.roads;
```

Data Output | Explain | Messages | History

	sum double precision
1	10064.0011938655

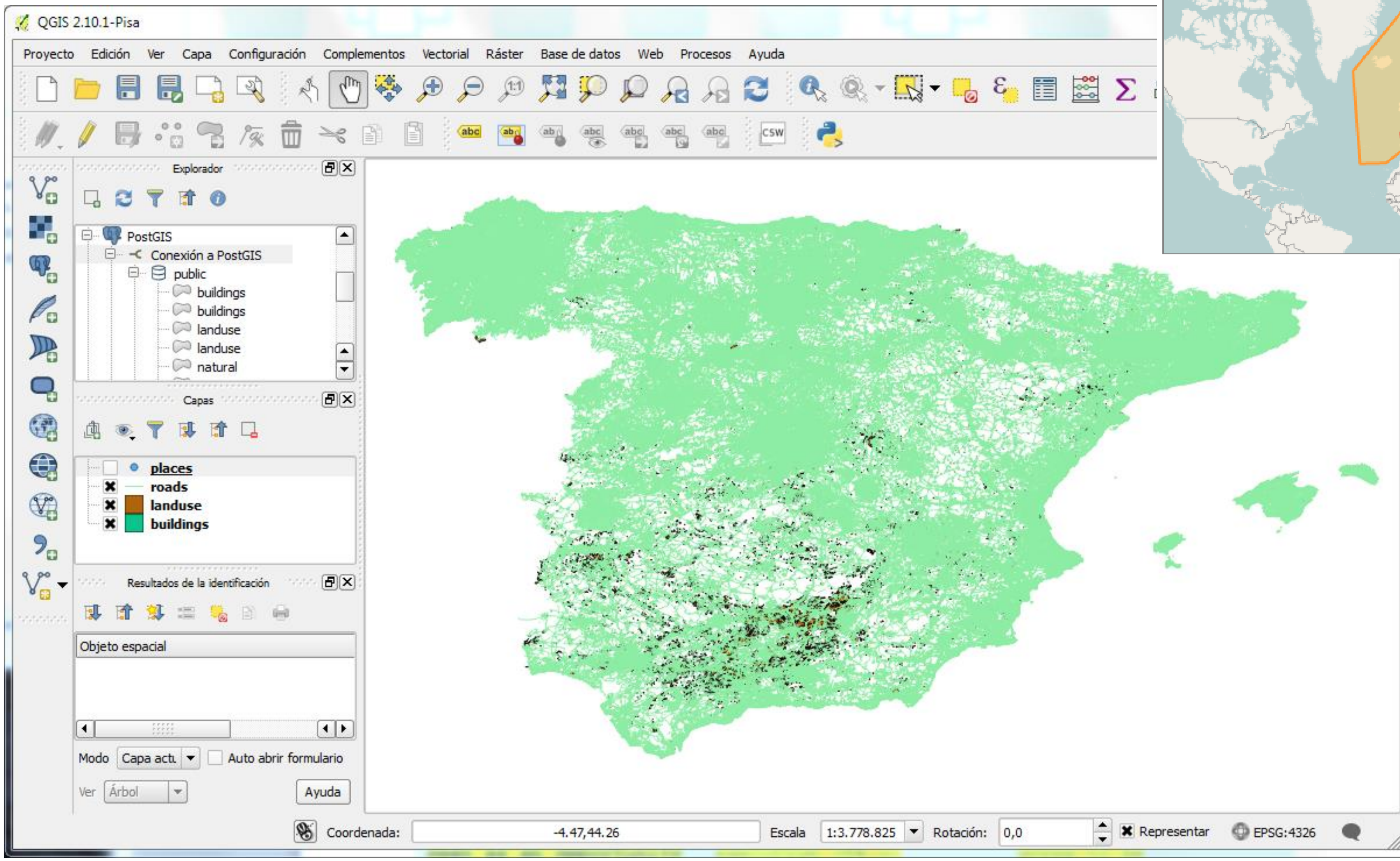
Area of schools

```
SELECT *, ST_Area(geom)
FROM public.buildings
WHERE type = 'school';
```

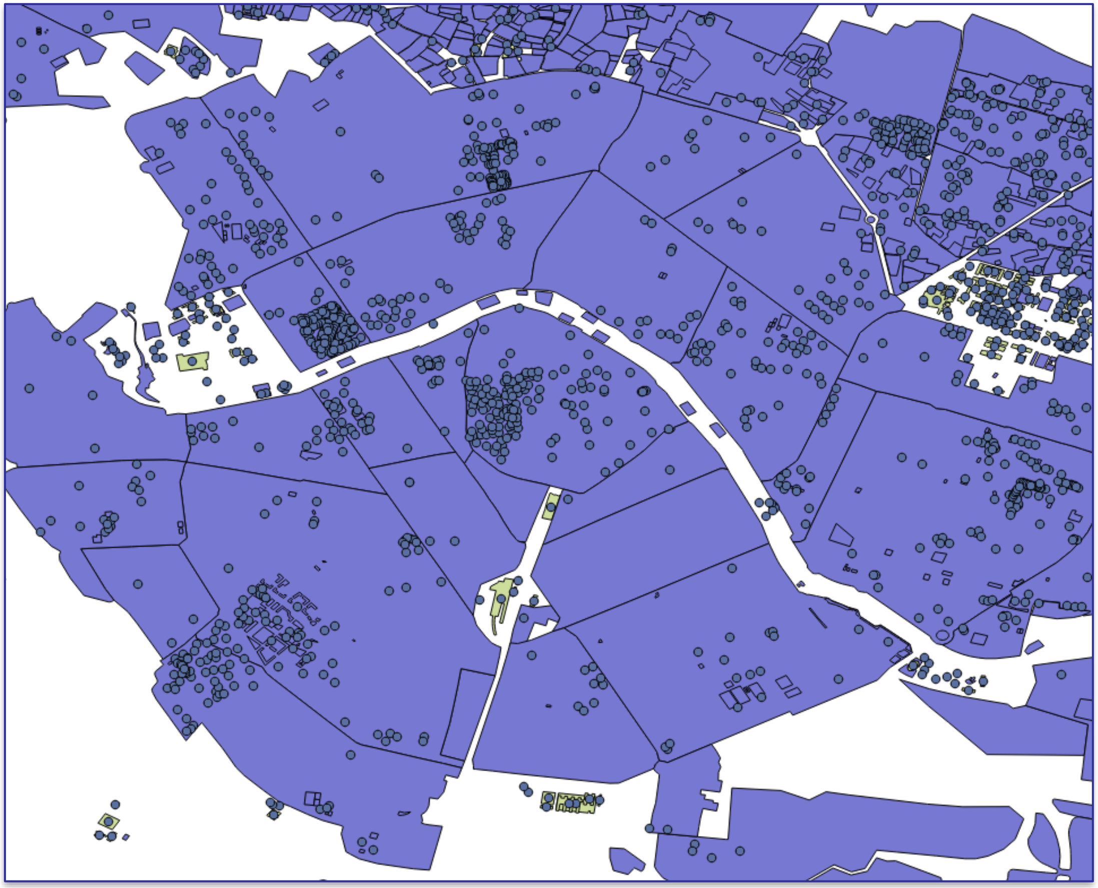
Data Output | Explain | Messages | History

gid integer	osm_id character varying(11)	name character varying(48)	type character varying(16)	geom geometry(MultiPolygon)	st_area double precision
216811	175995742	Colegio Bembibre	school	0106000000010000000103000000	1.35818285499596e-006
217400	176218951		school	0106000000010000000103000000	1.43345600005418e-008
217405	176221790		school	0106000000010000000103000000	2.13901980000518e-007
217407	176221793		school	0106000000010000000103000000	3.99283299998541e-008
217412	176252447	ESCI - Universitat Pompeu Fal	school	0106000000010000000103000000	1.27370539999921e-007
217804	176409101		school	0106000000010000000103000000	2.82516124998732e-007
218030	176505320		school	0106000000010000000103000000	6.38193199997864e-008

QGIS with .shp (shape) information retrieved from OpenStreetMaps



A GIS tool offers different **processing techniques & algorithms** for spatial analysis – very **useful** for **taking decisions**



E.g. **finding a good location** for building our delivery agencies by means of the calculus of **centroids** in landuses and buildings of Valencia