Emissions

Emissions

Every travelling vehicle creates some emissions:

- noise
- particulate matter
- gaseous: CO_2, CO, NO_2, ...

Amount depends on:

- vehicle type
- vehicle condition
- distance travelled
- road condition
- ...

Calculating Emissions

Basic idea: Track vehicle position and engine condition to calculate emissions.

We'll need EventHandlers:

- VehicleEntersTrafficEventHandler
  → engine get's started
- VehicleLeavesTrafficEventHandler
  → engine get's switched off, starts cooling down
- LinkLeaveEventHandler
  → vehicle has travelled along a link (except first link)

Calculating Emissions

```
public class VehicleTracker
        implements VehicleEntersTrafficEventHandler, VehicleLeavesTrafficEventHandler, LinkLeaveEventHandler {

    private Map<Id<Vehicle>, VehicleData> vehicleData = new HashMap<>();
    @Inject private Network network;

    @Override
    public void handleEvent(VehicleEntersTrafficEvent event) {
      getVehicleData(event.getVehicleId()).depart(event.getTime(), getLink(event.getLinkId()));
    }

    @Override
    public void handleEvent(VehicleLeavesTrafficEvent event) {
      getVehicleData(event.getVehicleId()).arrive(event.getTime(), getLink(event.getLinkId()));
    }

    @Override
    public void handleEvent(LinkLeaveEvent event) {
      getVehicleData(event.getVehicleId()).travelledLink(event.getTime(), getLink(event.getLinkId()));
    }

    private VehicleData getVehicleData(Id<Vehicle> vehicleId) {
      return this.vehicleData.computeIfAbsent(vehicleId, id -> new VehicleData());
    }

    private Link getLink(Id<Link> linkId) {
      return this.network.getLinks().get(linkId);
    }
}
```

Calculating Emissions

```
public class VehicleData {
  EngineState engineState = new EngineState(EngineState.COLD);
  boolean isFirstLink = false;

  public void depart(double time, Link link) {
    engineState.switchOn(time);
    isFirstLink = true;
  }

  public void arrive(double time, Link link) {
    engineState.switchOff(time);
  }

  public void travelledLink(double time, Link link) {
    if (!isFirstLink) {
      calculateEmissions(engineState, time, link);
    }
    isFirstLink = false;
  }
}

public class EngineState {
  // ....
}

public void calculateEmissions(...) {
  // ...
}
```

Calculating Emissions

Luckily, others did already most of the work! → **emissions contrib**

matsim/contribs/emissions, API Docs

Originally developed by Benjamin Kickhöfer and Friederike Hülsmann.

> This package provides a tool for exhaust emission calculation based on the "Handbook on Emission Factors for Road Transport" (HBEFA), version 3.1.

Hülsmann, F.; Gerike, R.; Kickhöfer, B.; Nagel, K. & Luz, R. (2011).

. Proceedings of the Conference on "Luftqualität an Straßen", FGSV Verlag GmbH, pp. 144-166. ISBN: 978-3-941790-77-3.

Kickhöfer, B.; Hülsmann, F.; Gerike, R. & Nagel, K. (2013).

. Smart Transport Networks: Decision Making, Sustainability and Market structure, Ed. by T. Vanoutrive and A. Verhetsel. NECTAR Series on Transportation and Communications Networks Research. Edward Elgar Publishing Ltd, pp. 180-207. ISBN: 978-1-78254-832-4.

Emissions Contrib

Based on various additional inputs, this extension calculates for each link and each vehicle, the exhaust emissions of that vehicle on that link:

```
<event time="10998.0" type="actend" person="pv_car_5315_9162_1" link="link36" actType="home"  />
<event time="10998.0" type="departure" person="pv_car_5315_9162_1" link="link36" legMode="car"  />
<event time="10998.0" type="PersonEntersVehicle" person="pv_car_5315_9162_1" vehicle="pv_car_5315_9162_1"  />
<event time="10998.0" type="vehicle enters traffic" person="pv_car_5315_9162_1" link="link36" vehicle="pv_car_5315_9162
<event time="10998.0" type="coldEmissionEvent" linkId="link36" vehicleId="pv_car_5315_9162_1"
       HC="3.78" NO2="0.00273337378166616" PM="0.00789998099207878" NMHC="3.57" NOX="0.33" CO="19.99" FC="23.79"  />
<event time="10999.0" type="left link" vehicle="pv_car_5315_9162_1" link="link36"  />
<event time="10999.0" type="entered link" vehicle="pv_car_5315_9162_1" link="link65"  />
<event time="11021.0" type="warmEmissionEvent" linkId="link65" vehicleId="pv_car_5315_9162_1"
       NO2="0.024" CO2_TOTAL="65.304" NOX="0.148" SO2="3.318369854241608E-4" HC="0.028000000000000004" CO="0.3680000000
<event time="11021.0" type="left link" vehicle="pv_car_5315_9162_1" link="link65"  />
<event time="11021.0" type="entered link" vehicle="pv_car_5315_9162_1" link="link52"  />
<event time="11021.0" type="warmEmissionEvent" linkId="link65" vehicleId="pv_car_5315_9162_1"
       CO2_TOTAL="65.58645338596195" CO="0.36878074213979445" FC="20.825753479050242" HC="0.028191202156684345" NO2="0.
<event time="11036.0" type="warmEmissionEvent" linkId="link52" vehicleId="pv_car_5315_9162_1"
       NO2="0.012" CO2_TOTAL="32.652" NOX="0.074" SO2="1.659184927120804E-4" HC="0.014000000000000002" CO="0.1840000000
<event time="11036.0" type="left link" vehicle="pv_car_5315_9162_1" link="link52"  />
<event time="11036.0" type="entered link" vehicle="pv_car_5315_9162_1" link="link25"  />
<event time="11036.0" type="warmEmissionEvent" linkId="link52" vehicleId="pv_car_5315_9162_1"
       CO2_TOTAL="33.1560402360515" CO="0.18539324034334764" FC="10.528165772532189" HC="0.014341201716738198" NO2="0.0
<event time="11046.0" type="vehicle leaves traffic" person="pv_car_5315_9162_1" link="link25" vehicle="pv_car_5315_9162
<event time="11046.0" type="PersonLeavesVehicle" person="pv_car_5315_9162_1" vehicle="pv_car_5315_9162_1"  />
<event time="11046.0" type="arrival" person="pv_car_5315_9162_1" link="link25" legMode="car"  />
<event time="11046.0" type="actstart" person="pv_car_5315_9162_1" link="link25" actType="work"  />
```

Emissions Contrib

The emissions extension currently calculates the following exhaust emissions:

- PM (particulate matter)
- NO_2 (nitrogen dioxide)
- NO_X (nitrogen oxides)
- CO_2 (carbon dioxide)
- CO (carbon monoxide)
- HC (hydro carbon)
- FC (?)
- NMHC (non-methane hydrocarbons)
- SO_2 (sulfur dioxide)

All values are reported in gram. For some types, this results in very small numbers.

Emissions Contrib

To use the emissions extension, add it as dependency to `pom.xml`:

```
<dependencies>
  <dependency>
    <groupId>org.matsim.contrib</groupId>
    <artifactId>emissions</artifactId>
    <version>${matsim.version}</version>
  </dependency>
</dependencies>
```

Calculate Emissions after Simulation

```
public class RunEmissionsOfflineExample {

  public static void main(String[] args) {
    Config config = ConfigUtils.loadConfig("/path/to/config_detailed.xml", new EmissionsConfigGroup());
    final Scenario scenario = ScenarioUtils.loadScenario(config);

    final EventsManager eventsManager = EventsUtils.createEventsManager();
    AbstractModule module = new AbstractModule() {
      public void install() {
        this.bind(Scenario.class).toInstance(scenario);
        this.bind(EventsManager.class).toInstance(eventsManager);
        this.bind(EmissionModule.class);
      }
    };
    Injector injector = org.matsim.core.controler.Injector.createInjector(config, module);
    EmissionModule emissionModule = injector.getInstance(EmissionModule.class);

    EventWriterXML emissionEventWriter = new EventWriterXML("path/to/output_emission_events.xml.gz");
    emissionModule.getEmissionEventsManager().addHandler(emissionEventWriter);

    MatsimEventsReader matsimEventsReader = new MatsimEventsReader(eventsManager);
    matsimEventsReader.readFile("/path/to/events.xml.gz"); // existing events file as input

    emissionEventWriter.closeFile();
  }
}
```

for up-to-date version, search for the class `RunDetailedEmissionToolOfflineExample`

Calculate Emissions during Simulation

```
public class RunEmissionsOnlineExample {

  public static void main(String[] args) {
    Config config = ConfigUtils.loadConfig("/path/to/config_detailed.xml", new EmissionsConfigGroup());
    Scenario scenario = ScenarioUtils.loadScenario(config);
    Controler controler = new Controler(scenario);

    controler.addOverridingModule(new AbstractModule() {
      @Override
      public void install() {
        bind(EmissionModule.class).asEagerSingleton();
      }
    });

    controler.run();
  }
}
```

for up-to-date version, search for the class `RunEmissionToolOnlineExample`

Analyze Emissions

```
EmissionGridAnalyzer gridAnalyzer = new EmissionGridAnalyzer.Builder()
  .withNetwork(scenario.getNetwork())
  .withTimeBinSize(3600)
  .withGridSize(10)
  .withSmoothingRadius(100)
  .withGridType(EmissionGridAnalyzer.GridType.Hexagonal)
  .build();

// output to JSON
gridAnalyzer.processToJsonFile("/path/to/output_emission_events.xml.gz", "/path/to/output_emission_grid.json");

// no output, but object to work with the data
TimeBinMap<Grid<Map<Pollutant, Double>>> map = gridAnalyzer.process("/path/to/output_emission_events.xml.gz");
```

Note: There is currently a bug that prevents the correct functioning. Need to replace `NOx` with `NOX` in the events.
Was fixed this week: [PR #601]

Analyze Emissions

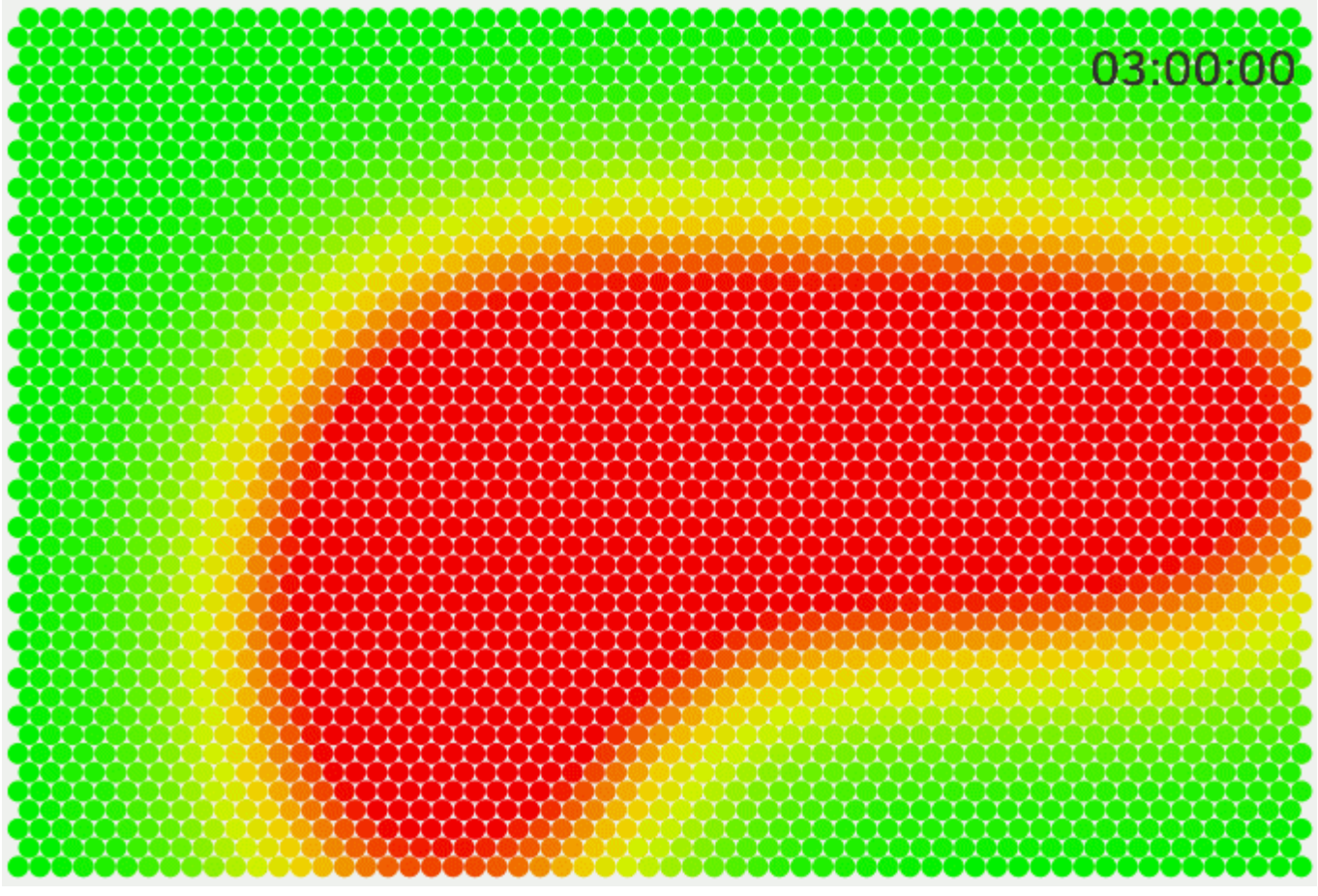
Writing pollution data as CSV for visualization in Via:

```
// ...continued

TimeBinMap<Grid<Map<Pollutant, Double>>> map = gridAnalyzer.process("/path/to/output_emission_events.xml.gz");

try (BufferedWriter writer = IOUtils.getBufferedWriter("/data/vis/tutorial-emissions/output/emissions.csv")) {
  writer.write("TIME,ID,X,Y,POLLUTANT,VALUE\n");
  for (TimeBinMap.TimeBin<Grid<Map<Pollutant, Double>>> timebin : map.getTimeBins()) {
    double startTime = timebin.getStartTime();
    Grid<Map<Pollutant, Double>> grid = timebin.getValue();
    for (Grid.Cell<Map<Pollutant, Double>> cell : grid.getCells()) {
      Coordinate coord = cell.getCoordinate();
      Map<Pollutant, Double> pollutants = cell.getValue();
      for (Map.Entry<Pollutant, Double> e : pollutants.entrySet()) {
        Pollutant pollutant = e.getKey();
        String id = (int) coord.x + "_" + (int) coord.y + "_" + pollutant.name();
        writer.write(startTime +","+ id +","+ coord.x +","+ coord.y +","+ pollutant.name() +","+ e.getValue() +"\n");
      }
    }
  }
}
```

Analyze Emissions



Required Input for Emissions: Config

Additional config group in `config.xml`:

```
<module name="emissions" >

  <!-- REQUIRED  file with HBEFA 3.1 fleet average cold emission factors -->
  <param name="averageFleetColdEmissionFactorsFile" value="sample_EFA_ColdStart_vehcat_2005average.txt" />

  <!-- REQUIRED  file with HBEFA 3.1 fleet average warm emission factors -->
  <param name="averageFleetWarmEmissionFactorsFile" value="sample_EFA_HOT_vehcat_2005average.txt" />

  <!-- OPTIONAL  file with HBEFA 3.1 detailed cold emission factors -->
  <param name="detailedColdEmissionFactorsFile" value="sample_EFA_ColdStart_SubSegm_2005detailed.txt" />

  <!-- OPTIONAL  file with HBEFA 3.1 detailed warm emission factors -->
  <param name="detailedWarmEmissionFactorsFile" value="sample_EFA_HOT_SubSegm_2005detailed.txt" />

  <!-- "fromLinkAttributes" will eventually become default:-->
  <param name="hbefaRoadTypeSource" value="fromLinkAttributes" />

  <!-- if true then detailed emission factor files must be provided! -->
  <param name="usingDetailedEmissionCalculation" value="true" />
</module>
```

Required Input for Emissions: Emission Factors

These data should be exported from HBEFA.

The API Docs give detailed instructions.

Export from HBEFA 3.x

Currently, data from the HBEFA 3.x Microsoft Access database needs to be exported manually. Consequently, the following steps within HBEFA 3.x need still be done manually:

- Install and open HBEFA 3.x
- Select your country and language
- Go to "CaseDefinition" > "New"
- Select the desired parameters:
  - VEHICLE CATEGORIES: Currently, PC (passenger car), HGV (heavy goods vehicle), MOTORCYCLE and ZEV (zero emission vehicle) are supported (see `HbefaVehicleCategory`)
  - COMPONENTS: MATSim will handle any components that are exported from the hbefa database
  - YEARS: Choose the year of your scenario (only when exporting the mandatory average emission factors files)
  - FLEET COMPOSITION: Choose "EF weighted with fleet composition" for the mandatory average emission factors files, and "EF per subsegment (without weighting)" for the optional detailed emission factors files
  - HOT EMISSION FACTORS: Choose "Individual TrafficSituations" > "Construct your own list" > "SelectAll" > "Return"
  - COLD START EXCESS EMISSION FACTORS: Tick this option and choose "Construct your own list" > select all "patterns" with average temperature, detailed parking time (0-1h .. >12h), and detailed distance (0-1km and 1-2km) > "Return"
  - Leave everything else as default
- Enter "Name of parameter set" and press "Calculate"
- Save the two generated tables using "Results" > "Export" to the desired location

All these emission factor files need to be converted into *.txt or *.csv with ";" as delimiter. Their column headers should automatically match the parser definition in the respective method of the `EmissionModule`.

Sample data can be downloaded from here.

Required Input for Emissions: Road Types

HBEFA contains data for different road types. Thus, each link in MATSim must be assigned a matching road type.

Easiest solution is to add them as link attributes:

```
<network>
  <nodes>
    ...
  </nodes>

  <links capperiod="16:00:00" effectivecellsize="7.5" effectivelanewidth="NaN">
    <link id="link14" from="node1" to="node4" length="200.0" freespeed="9.722222222222221" capacity="8400.0" permlanes=
      <attributes>
        <attribute name="hbefa_road_type" class="java.lang.String" >URB/Local/50</attribute>
      </attributes>
   </link>
   ...
  </links>
</network>
```

Required Input for Emissions: Vehicles

The emissions extension uses the default vehicles data structure to specify vehicles' attributes.

`config.xml`:

```
<module name="vehicles" >
  <param name="vehiclesFile" value="emissionVehicles.xml" />
</module>
```

The description of vehicleTypes needs to follow a special format to encode the attributes required for the emission calculation:

Example:

```
BEGIN_EMISSIONSPASSENGER_CAR;petrol (4S);>=2L;PC-P-Euro-1END_EMISSIONS
\_____________/\___________/ \_________/ \__/ \_________/\___________/
     Header       HBEFA-    ;Technology ;Size-; EmConcept  End-Marker
                 Vehicle-                class
                 Category
```

Technology, Size-class and EmConcept are only required if the detailed emission calculation is enabled.

Required Input for Emissions: Vehicles

```
<?xml version="1.0" encoding="UTF-8"?>
<vehicleDefinitions xmlns="http://www.matsim.org/files/dtd"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.matsim.org/files/dtd http://www.matsim.org/files/dtd/vehicleDefinitions_v1.0.xsd">
    <vehicleType id="car_average">
        <description>
            BEGIN_EMISSIONSPASSENGER_CAR;average;average;averageEND_EMISSIONS
        </description>
        <length meter="7.5"/>
        <width meter="1.0"/>
    </vehicleType>
    <vehicleType id="car_petrol">
        <description>
            BEGIN_EMISSIONSPASSENGER_CAR;petrol (4S);&gt;=2L;PC-P-Euro-1END_EMISSIONS
        </description>
        <length meter="7.5"/>
        <width meter="1.0"/>
    </vehicleType>
    <vehicleType id="car_diesel">
        <description>
            BEGIN_EMISSIONSPASSENGER_CAR;diesel;&lt;1,4L;PC-D-Euro-3END_EMISSIONS
        </description>
        <length meter="7.5"/>
        <width meter="1.0"/>
    </vehicleType>
    <vehicleType id="truck">
        <description>
            BEGIN_EMISSIONSHEAVY_GOODS_VEHICLE;average;average;averageEND_EMISSIONS
        </description>
        <length meter="7.5"/>
        <width meter="1.0"/>
```

Adapting Network for Emissions

Adding a `hbefa_road_type` attribute to each link:

```
public class AddRoadtype {

  public void run(String inputNetworkFilename, String outputNetworkFilename) {
    Network network = NetworkUtils.createNetwork();
    new MatsimNetworkReader(network).readFile(inputNetworkFilename);

    for (Link link : network.getLinks().values()) {
      if (link.getFreespeed() <= (50 / 3.6)) {
        link.getAttributes().putAttribute("hbefa_road_type", "URB/Local/50");
      } else {
        link.getAttributes().putAttribute("hbefa_road_type", "URB/Local/80"); // TODO adapt to real hbefa values
      }
    }

    new NetworkWriter(network).write(outputNetworkFilename);
  }

  public static void main(String[] args) {
    new AddRoadtype().run(
        "/path/to/network.xml.gz",
        "/path/to/output/network.xml.gz"
    );
  }

}
```

Adapting Network for Emissions

If our network was created by pt2matsim, there exists code to calculate the correct HBEFA road type: See class `org.matsim.contrib.emissions.OsmHbefaMapping`.

This class is not directly accessible, but it can be enabled to be used in the config:

```
<module name="emissions" >
  <!-- "fromLinkAttributes" will eventually become default:-->
  <param name="hbefaRoadTypeSource" value="fromOsm" />
</module>
```

But it's usage is discourage...

Creating Vehicles

```
public class CreateEmissionVehicles {
  public void run(String inputPoopulationFilename, String outputVehiclesFilename) {
    Scenario scenario = ScenarioUtils.createScenario(ConfigUtils.createConfig());
    new PopulationReader(scenario).readFile(inputPoopulationFilename);

    Vehicles vehicles = scenario.getVehicles();
    VehiclesFactory factory = vehicles.getFactory();

    VehicleType avgCarType = factory.createVehicleType(Id.create("car_average", VehicleType.class));
    avgCarType.setDescription("BEGIN_EMISSIONSPASSENGER_CAR;average;average;averageEND_EMISSIONS");
    vehicles.addVehicleType(avgCarType);

    VehicleType petrolCarType = factory.createVehicleType(Id.create("car_petrol", VehicleType.class));
    petrolCarType.setDescription("BEGIN_EMISSIONSPASSENGER_CAR;petrol (4S);>=2L;PC-P-Euro-1END_EMISSIONS");
    vehicles.addVehicleType(petrolCarType);

    Random r = new Random(123456);
    for (Person person : scenario.getPopulation().getPersons().values()) {
      VehicleType vehType = avgCarType;
      if (r.nextDouble() < 0.5) {
        vehType = petrolCarType;
      }
      Vehicle vehicle = factory.createVehicle(Id.create(person.getId().toString(), Vehicle.class), vehType);
      vehicles.addVehicle(vehicle);
    }

    new VehicleWriterV1(vehicles).writeFile(outputVehiclesFilename);
  }
}
```

Electric Vehicles

Electric Vehicles

What could be of interest?

- Energy consumption while driving
- Charging behaviour while being parked
- Charging stops on long-distance trips
- Charging breaks in commercial fleet (e.g. taxi)
- Use of vehicle's charge for auxiliary reasons

Electric Vehicles

What data do we need?

- Charging Stations
  - Locations
  - Types (provided power, plug types, number of vehicles that can be charged)
- Fleet of electric vehicles
  - Battery capacity
  - Initial charge
  - Power consumption while driving
- other data?

Electric Vehicles

How to track energy consumption while driving?

EventHandler tracking vehicles (LinkEnterEvent, LinkLeaveEvent)

How to track vehicle charging?

EventHandler tracking parked vehicles (VehicleLeavesTrafficEvent, VehicleEntersTrafficEvent)

Does every parked car charge when close to a charger? How do we know if a vehicle charges? Does it free the plug once if is fully charged?

How to decide where to charge on long-distance trips?

Pre-plan it using a specialized router, or dynamically re-planing during the simulation.

Others have already done some of that work for us → **ev contrib**

Electric Vehicles contrib

matsim/contribs/ev

Mostly maintained by Michał Maciejewski (TU Berlin) and Joschka Bischoff (TU Berlin, soon SBB), Tilmann Schlenther (TU Berlin)

Bischoff, J. et al. (2019).

. Link

Reza Vosooghi, Jakob Puchinger, Joschka Bischoff, Marija Jankovic, Anthony Vouillon.

2019. hal-

02136507 Link

Electric Vehicles contrib

The contrib currently supports:

- Specify type and location of chargers
- Keep track of a EV fleet
- Pre-plan stops on long distance trips for charging

The contrib currently **does not** support:

- deciding when to charge when used for short-distance trips
- parking-search for short-distance trips
- deciding when to charge (at home? below power-treshold?, always when possible?)

(some of these features might be supported in the future.)

Required Input: Charger Locations

Chargers are located on links, and can provide power to a number of vehicles:

```
<!DOCTYPE chargers SYSTEM "http://matsim.org/files/dtd/chargers_v1.dtd">
<chargers>
    <charger id="charger1" link="4696241_0" power="100.0" capacity="5" type="default" />
    <charger id="charger2" link="299552374_0" power="100.0" capacity="5" type="default" />
    <charger id="charger3" link="150062610_0" power="100.0" capacity="5" type="default" />
</chargers>
```

`power` is per plug, in kW.
It is possible to have multiple chargers per link.

If `type` is omitted, `default` will be used.
`capacity` might be renamed to `plugCount` in the future, `power` might be renamed to `maxPower` in the future.

Required Input: Electric Vehicle Fleet

```
<!DOCTYPE vehicles SYSTEM "http://matsim.org/files/dtd/electric_vehicles_v1.dtd">
<vehicles>
    <vehicle id="0" battery_capacity="40" initial_soc="40" chargerTypes="default" vehicleType="defaultVehicleType"/>
    <vehicle id="1" battery_capacity="40" initial_soc="40" chargerTypes="default" vehicleType="defaultVehicleType"/>
    <vehicle id="2" battery_capacity="40" initial_soc="40" chargerTypes="default" vehicleType="defaultVehicleType"/>
    <vehicle id="3" battery_capacity="40" initial_soc="40" chargerTypes="default" vehicleType="defaultVehicleType"/>
    <vehicle id="4" battery_capacity="40" initial_soc="40" chargerTypes="default" vehicleType="defaultVehicleType"/>
</vehicles>
```

`chargerTypes` is a list of comma-separated types. If empty, `default` will be used.
If `vehicleType` is omitted, `defaultVehicleType` will be used.

Electric Vehicles contrib

To use it, switch version in pom.xml to `12.0-2019w27-SNAPSHOT` and add the dependencies:

```
<dependencies>
    <dependency>
        <groupId>org.matsim.contrib</groupId>
        <artifactId>ev</artifactId>
        <version>${matsim.version}</version>
    </dependency>
    <dependency>
        <groupId>org.matsim.contrib</groupId>
        <artifactId>dvrp</artifactId>
        <version>${matsim.version}</version>
    </dependency>
</dependencies>

<properties>
     <matsim.version>12.0-2019w27-SNAPSHOT</matsim.version>
</properties>
```

Electric Vehicles contrib

Main class:

```
public void run(URL configUrl) {
  Config config = ConfigUtils.loadConfig(configUrl, new EvConfigGroup());
  Scenario scenario = ScenarioUtils.loadScenario(config);
  Controler controler = new Controler(scenario);
  controler.addOverridingModule(new EvModule());
  controler.addOverridingModule(new AbstractModule() {
    @Override
    public void install() {
      addRoutingModuleBinding(TransportMode.car).toProvider(new EvNetworkRoutingProvider(TransportMode.car));
      installQSimModule(new AbstractQSimModule() {
        @Override
        protected void configureQSim() {
          bind(VehicleChargingHandler.class).asEagerSingleton();
        }
      });
    }
  });
  controler.configureQSimComponents(components -> components.addNamedComponent(EvModule.EV_COMPONENT));

  controler.run();
}
```

Electric Vehicles contrib

`EvNetworkRoutingProvider` calculates routes for EV vehicles, and pre-plans stops on long distance trips to re-charge:

```
<person id="0">
  <plan selected="yes">
    <activity type="h" link="22760964_0" x="290625.48113156157" y="5885518.622230196" end_time="08:20:05" >
    </activity>
    <leg mode="car" dep_time="08:20:05" trav_time="01:23:52">
      <route type="links" start_link="22760964_0" end_link="4696241_0" trav_time="01:23:52" distance="152302.0283119321
    </leg>
    <activity type="car charging" link="4696241_0" x="405535.4410234112" y="5828853.337960448" max_dur="00:30:00" >
    </activity>
    <leg mode="car" dep_time="10:13:57" trav_time="01:23:50">
      <route type="links" start_link="4696241_0" end_link="256204618_0" trav_time="01:23:50" distance="170412.506861745
    </leg>
    <activity type="h" link="256204618_0" x="468532.721161525" y="5728751.474117422" >
    </activity>
  </plan>
</person>
```

EV specific Events

- ChargingStartEvent

```
<event time="33129.0" type="charging_start" charger="charger3" vehicle="12" chargerType="default"  />
```

- ChargingEndEvent

```
<event time="35829.0" type="charging_end" charger="charger3" vehicle="12"  />
```

Electric Vehicles contrib

The contrib provides many interfaces where custom implementations could be used:

Examples:

- ChargingLogic
- BatteryCharging
- DriveEnergyConsumption.Factory
- TemperatureService
- AuxEnergyConsumption.Factory
- AuxDischargingHandler.VehicleProvider
- ChargingLogic.Factory
- ChargingPower.Factory (ChargingPower is a superinterface to BatteryCharging)

Electric Vehicles contrib

(My personal impression:)

- There is a lot of useful infrastructure
  E.g. ChargingLogic, TemperatureService, BatteryCharging
- This infrastructure is currently only used for one single use-case: long distance trips
- There are many other use cases which are not yet covered by the extension
- The extension is rather new and a lot of development is currently happening,
  things might be different very soon!

Code Examples

Filter a network by time limit

```
public class FilterNetworkByTime {
  public void run(String inputNetworkFilename, String outputNetworkFilename) {
    Network network = NetworkUtils.createNetwork();
    new MatsimNetworkReader(network).readFile(inputNetworkFilename);

    FreespeedTravelTimeAndDisutility freespeed = new FreespeedTravelTimeAndDisutility(-6/3600, 6/3600, 0);
    LeastCostPathTree tree = new LeastCostPathTree(freespeed, freespeed);
    Node origin = network.getNodes().get(Id.create("1378778663", Node.class));
    tree.calculate(network, origin, 15*60);
    Map<Id<Node>, LeastCostPathTree.NodeData> reachableNodes = tree.getTree();

    Set<Link> linksToRemove = new HashSet<>();
    for (Link link : network.getLinks().values()) {
      boolean fromNodeReachable = reachableNodes.containsKey(link.getFromNode().getId());
      boolean toNodeReachable = reachableNodes.containsKey(link.getToNode().getId());
      if (!fromNodeReachable && !toNodeReachable) {
        linksToRemove.add(link);
      }
    }

    for (Link link : linksToRemove) {
      network.removeLink(link.getId());
    }

    // TODO run network cleaner...
    new NetworkWriter(network).write(outputNetworkFilename);
  }
}
```

Thank you!

Marcel Rieser
Simunto GmbH
rieser@simunto.com