ItineraryScheduling (Appointment insertedAppointment){

ArrayList<Itinerary> feasibleItineraries = new ArrayList<Itinerary>;

feasibleVehicles = PermittedVehicles(feasibleVehicles);

for (Vehicle vehicle : feasibleVehicles){

Itinerary permittedItinerary = computeShortTrack(vehicle, insertedAppointment.predecessor.location, insertedAppointment); //Google API

if(isCoherent(vehicle, avoidableTimeForMOP, maximumDistanceForMOP, maximumCostForItineray, permittedItinerary) &&

ConsistencyCheck (permittedItinerary, insertedAppointment.priority)){

feasibleItineraries.add(permittedItinerary);

}

}

}

ProposeItinerary (feasibleItineraries);

}

ProposeItinerary(ArrayList<Itineray> feasibleItineraries){

Itineray proposedItineary = new Array[5];

proposedItineary[0] = minTime(feasibleItineraries);

proposedItineary[1] = eco(feasibleItineraries);

proposedItineary[2] = minCost(feasibleItineraries);

proposedItineary[3] = minChange(feasibleItineraries);

proposedItineary[4] = minWalkDist(feasibleItineraries);

PersonalVehicleCheck(proposedItineary);

}

PersonalVehicleCheck(Array proposedItineary){

for (Itinerary itinerary : proposedItineary){

if(personalVehicles.contains(itinerary.vehicle)

return proposedItineary;

}

proposedItineary[0] = computeShortTrack (personalVehicles(0), insertedAppointment.predecessor.location, insertedAppointment);

PermittedVehicles (Vehicle feasibleVehicles){

for (Vehicle vehicle : personalVehicles){//in the feasible vehicles we consider all the personal vehicle

feasibleVehicles.add(personalVehicles);

}

EcologicCondition(ecologist, feasibleVehicles);

WeatherCondition(feasibleVehicles, weatherForecast);

AppointmentTypeCondition(feasibleVehicles, insertedAppointmentType);

}

minTime (ArryList<Itinerary> feasibleItineraries){

Itinerary choice = feasibleItineraries(0);

for (Itinerary itinerary : feasibleItineraries){

if (itinerary.time < choice.time)

choice = itinerary;

}

return choice;

}

WeatherCondition (ArrayList<Vehicle> feasibleVehicles, Weather weatherForecast) {

if (weatherForecast.POP >= 40 || weatherForecast.temperature <= 18){

feasibleVehicles.remove(bike);

feasibleVehicles.remove(foot);

}

}

AppointmentTypeCondition (ArrayList<Vehicle> feasibleVehicles, AppointmentType insertedAppointmentType){//Avoid vehicles becuase of appointment type

for (Vehicle vehicle : feasibleVehicles){

if (insertedAppointmentType.avoidedVehicles.contains(vehicle)){

feasibleVehicles.remove(vehicle);

EcologicCondition (boolean ecologist, feasibleVehicles)

//TODO

isCoherent (int maxWalkDist, int maxCost, Itinerary itinerary){//no exceed maxWalkDist and maxCost

if (itinerary.walkDist < maxWalkDist && itinerary.cost < maxCost)

return true;

return false;

ConsistencyCheck (Itinerary permittedItinerary, int priority, Appointment insertedAppointment){

for (DailySchedule dailySchedule : dailySchedules){

if (dailySchedule.time == insertedAppointment.time){

Array appointments = dailySchedule.appointments;

for (Appointment appointment : appointments){

if (appointment.start <= insertedAppointment.start <= appointment.end ||

appointment.start <= insertedAppointment <= appointment.end)

return false;

}

}

}

return true;

}