VRPTWOptimizer

Generated by Doxygen 1.9.3

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 VRPTWOptimizer Namespace Reference

Namespaces

- namespace DistanceProviders
- namespace Dto
- namespace Enums
- namespace Interfaces
- namespace Logging

Classes

- class CargoUnit

  Describes a single position on an order list (depending on the context it could be a europallet or single type of product with its quantity)
- class Driver

  Represents the truck/tractor driver
- class InsertionResult

  Description of TransportRequest insert consequences
- class Resource

  Generalized time bound resource (driver, machine, vehicle)
- class TransportRequest

  Description of the request to move cargo from pickup to delivery Location
- class Vehicle

  Defines properties of a Vehicle
- class VRPCostFunction

  Class for calculating solution costs
- class VRPDefinition

  Describes data for a generalized Vehicle Routing Problem
- class VRPOptimizerResult

  Represents the output of Vehicle Routing Problem optimization algorithm
- class VRPSolution

  Definition of structure describing solution to the Vehicle Routing Problem. Includes Vehicle assignment to TransportRequest and Vehicle schedule

5.2 VRPTWOptimizer.DistanceProviders Namespace Reference

Classes

- class DictionaryDistanceProviderBase

5.3 VRPTWOptimizer.Dto Namespace Reference

Classes

- class PickingSchedule

  Complete picking schedule for one day
- class StorePickingList

  Picking order details for single store
- class TimeInterval

  Represents time interval for vehicle yard availability
- class TransportPickingLists

  Picking orders for single transport
- class VehicleSchedule

  Schedules vehicle presence at warehouse grounds

5.4 VRPTWOptimizer.Enums Namespace Reference

Enumerations

- enum Aggregation { Sum = 1 , Max = 2 }

  Enumerator describing available types of cargo size aggregation
- enum CargoType { Food = 1 , EmptyBoxes = 2 , Garbage = 3 }

  Type of cargo that would determine if it can be transported with other types
- enum CargoUnitType { Box = 1 }

  Types of cargo units
- enum RequestType { GoodsDistribution = 1 , ContainerRetrieval = 2 , Backhauling = 3 }

  Type of TransportRequest possibly useful to set priorities

5.4.1 Enumeration Type Documentation

5.4.1.1 Aggregation

`enum VRPTWOptimizer.Enums.Aggregation`

Enumerator describing available types of cargo size aggregation

Enumerator

Sum	Sum aggregate is coded as 1
Max	Max aggregate is coded as 2

Definition at line 6 of file Aggregation.cs.

5.4.1.2 CargoType

`enum VRPTWOptimizer.Enums.CargoType`

Type of cargo that would determine if it can be transported with other types

Enumerator

Food	Food is encoded as 1
EmptyBoxes	Empty coolboxes and other containers are encoded as 2
Garbage	Garbage is encoded as 3 (it is assumed that food cannot be transported with garbage)

Definition at line 6 of file CargoType.cs.

5.4.1.3 CargoUnitType

`enum VRPTWOptimizer.Enums.CargoUnitType`

Types of cargo units

Enumerator

Box	Represents box of products (like a pack of 12 milks)

Definition at line 12 of file CargoUnitType.cs.

5.4.1.4 RequestType

`enum VRPTWOptimizer.Enums.RequestType`

Type of TransportRequest possibly useful to set priorities

Enumerator

GoodsDistribution	Delivering ordered cargo from warehouse to final location
ContainerRetrieval	Delivering reusable containers to warehouse
Backhauling	Getting goods from external entities (may constitute additional profit for company)

Definition at line 6 of file RequestType.cs.

5.5 VRPTWOptimizer.Interfaces Namespace Reference

Classes

- interface IRoute

  Represents route assigned to Vehicle
- interface ITimeEstimator

  Predicts time of loading and unloading cargo at Location
- interface IVRPOptimizer

  Optimizes Vehicle Routing Problem
- interface IVRPOptimizerFactory

  Provides new instance of optimizer (follows Abstract Factory desing pattern)
- interface IVRPProvider

  Provides problem data for IVRPOptimizer
- interface IVRPSolutionWriter
- class VRPResult

  Class containing list of routes for each tractor and straight truck

5.6 VRPTWOptimizer.Logging Namespace Reference

Classes

- class JSONDefinitionWriter

  Serializes solution to JSON

Chapter 6

Class Documentation

6.1 VRPTWOptimizer.CargoUnit Class Reference

Describes a single position on an order list (depending on the context it could be a europallet or single type of product with its quantity)

Properties

- string CargoGroup `[get, set]`
  Group of goods (e.g. fresh meat)
- CargoUnitType CargoUnitType `[get, set]`
  Smallest considered piece of cargo (e.g. piece, box, europallet)
- int GoodsId `[get, set]`
  Identifier of product
- string GoodsName `[get, set]`
  Name of product
- int Priority `[get, set]`
  The smaller the number the higher delivery priority
- double[ ] Size `[get, set]`
  Cargo size specified in the units used to compute transportation capacity
- int UnitsCount `[get, set]`
  Number of cargo units (e.g. 12 boxes)

6.1.1 Detailed Description

Describes a single position on an order list (depending on the context it could be a europallet or single type of product with its quantity)

Definition at line 14 of file CargoUnit.cs.

6.1.2 Property Documentation

6.1.2.1 CargoGroup

```
string VRPTWOptimizer.CargoUnit.CargoGroup  [get], [set]
```

Group of goods (e.g. fresh meat)

Definition at line 20 of file CargoUnit.cs.

6.1.2.2 CargoUnitType

```
CargoUnitType VRPTWOptimizer.CargoUnit.CargoUnitType  [get], [set]
```

Smallest considered piece of cargo (e.g. piece, box, europallet)

Definition at line 25 of file CargoUnit.cs.

6.1.2.3 GoodsId

```
int VRPTWOptimizer.CargoUnit.GoodsId  [get], [set]
```

Identifier of product

Definition at line 30 of file CargoUnit.cs.

6.1.2.4 GoodsName

```
string VRPTWOptimizer.CargoUnit.GoodsName  [get], [set]
```

Name of product

Definition at line 35 of file CargoUnit.cs.

6.1.2.5 Priority

```
int VRPTWOptimizer.CargoUnit.Priority  [get], [set]
```

The smaller the number the higher delivery priority

Definition at line 40 of file CargoUnit.cs.

6.1.2.6 Size

```
double [] VRPTWOptimizer.CargoUnit.Size  [get], [set]
```

Cargo size specified in the units used to compute transportation capacity

Definition at line 45 of file CargoUnit.cs.

6.1.2.7 UnitsCount

```
int VRPTWOptimizer.CargoUnit.UnitsCount  [get], [set]
```

Number of cargo units (e.g. 12 boxes)

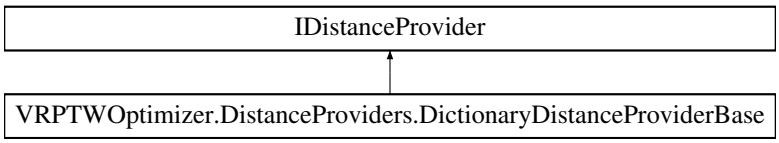
Definition at line 50 of file CargoUnit.cs.

The documentation for this class was generated from the following file:

- CargoUnit.cs

6.2 VRPTWOptimizer.DistanceProviders.DictionaryDistanceProviderBase Class Reference

Inheritance diagram for VRPTWOptimizer.DistanceProviders.DictionaryDistanceProviderBase:



Public Member Functions

- Distance GetDistance (Location from, Location to, VehicleRoadRestrictionProperties vehicleProperties)

Protected Member Functions

- DictionaryDistanceProviderBase (bool selfContain)
- void InitializeDistanceDictionary (List< Distance > distances)

Protected Attributes

- Dictionary< string, Dictionary< string, Dictionary< VehicleRoadRestrictionProperties, Distance > > > distanceMatrix
- Dictionary< VehicleRoadRestrictionProperties, VehicleRoadRestrictionProperties > vehicleToProfileMapper
- bool SelfContain

Properties

- List< Distance > StoredDistances [get, set]

6.2.1 Detailed Description

Definition at line 8 of file DictionaryDistanceProviderBase.cs.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 DictionaryDistanceProviderBase()

```
VRPTWOptimizer.DistanceProviders.DictionaryDistanceProviderBase.DictionaryDistanceProviderBase
(
            bool selfContain )  [protected]
```

Definition at line 14 of file DictionaryDistanceProviderBase.cs.

6.2.3 Member Function Documentation

6.2.3.1 GetDistance()

```
Distance VRPTWOptimizer.DistanceProviders.DictionaryDistanceProviderBase.GetDistance (
            Location from,
            Location to,
            VehicleRoadRestrictionProperties vehicleProperties )
```

Definition at line 45 of file DictionaryDistanceProviderBase.cs.

6.2.3.2 InitializeDistanceDictionary()

```
void VRPTWOptimizer.DistanceProviders.DictionaryDistanceProviderBase.InitializeDistance↩
Dictionary (
            List< Distance > distances )  [protected]
```

Definition at line 24 of file DictionaryDistanceProviderBase.cs.

6.2.4 Member Data Documentation

6.2.4.1 distanceMatrix

```
Dictionary<string, Dictionary<string, Dictionary<VehicleRoadRestrictionProperties, Distance>
> > VRPTWOptimizer.DistanceProviders.DictionaryDistanceProviderBase.distanceMatrix  [protected]
```

Definition at line 10 of file DictionaryDistanceProviderBase.cs.

6.2.4.2 SelfContain

```
bool VRPTWOptimizer.DistanceProviders.DictionaryDistanceProviderBase.SelfContain  [protected]
```

Definition at line 12 of file DictionaryDistanceProviderBase.cs.

6.2.4.3 vehicleToProfileMapper

```
Dictionary<VehicleRoadRestrictionProperties, VehicleRoadRestrictionProperties> VRPTWOptimizer.↩
DistanceProviders.DictionaryDistanceProviderBase.vehicleToProfileMapper  [protected]
```

Definition at line 11 of file DictionaryDistanceProviderBase.cs.

6.2.5 Property Documentation

6.2.5.1 StoredDistances

```
List<Distance> VRPTWOptimizer.DistanceProviders.DictionaryDistanceProviderBase.StoredDistances
[get], [set]
```

Definition at line 22 of file DictionaryDistanceProviderBase.cs.

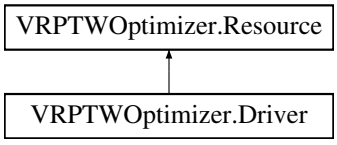
The documentation for this class was generated from the following file:

- DictionaryDistanceProviderBase.cs

6.3 VRPTWOptimizer.Driver Class Reference

Represents the truck/tractor driver

Inheritance diagram for VRPTWOptimizer.Driver:



Public Member Functions

- Driver (int id, double availabilityStart, double availabilityEnd, int[ ] compatibileVehiclesIds)

  Creates new Driver object

Properties

- int[ ] CompatibileVehiclesIds `[get, protected set]`

  List of Vehicle Ids that the driver is able and allowed to drive

6.3.1 Detailed Description

Represents the truck/tractor driver

Definition at line 6 of file Driver.cs.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 Driver()

```
VRPTWOptimizer.Driver.Driver (
            int id,
            double availabilityStart,
            double availabilityEnd,
            int[] compatibileVehiclesIds )
```

Creates new Driver object

Parameters

id	
availabilityStart	
availabilityEnd	
compatibileVehiclesIds	

Definition at line 20 of file Driver.cs.

6.3.3 Property Documentation

6.3.3.1 CompatibileVehiclesIds

```
int [] VRPTWOptimizer.Driver.CompatibileVehiclesIds  [get], [protected set]
```

List of Vehicle Ids that the driver is able and allowed to drive

Definition at line 11 of file Driver.cs.

The documentation for this class was generated from the following file:

- Driver.cs

6.4 VRPTWOptimizer.InsertionResult Class Reference

Description of TransportRequest insert consequences

Public Member Functions

- InsertionResult (double oldDistanceBetween, double newAddedDistance, double oldNextArriveTime, double newNextArriveTime, double expectedArriveTime, double maxDelay, bool success)

Properties

- double ExpectedArriveTime `[get]`
  Time when we will arrive to serve inserted TransportRequest
- double MaxDelay `[get]`
  Max delay generated by insertion of TransportRequest
- double NewAddedDistance `[get]`
  Route length added by insertion of TransportRequest
- double NewNextArriveTime `[get]`
- double OldDistanceBetween `[get]`
- double OldNextArriveTime `[get]`
- bool Success `[get]`

6.4.1 Detailed Description

Description of TransportRequest insert consequences

Definition at line 6 of file InsertionResult.cs.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 InsertionResult()

```
VRPTWOptimizer.InsertionResult.InsertionResult (
            double oldDistanceBetween,
            double newAddedDistance,
            double oldNextArriveTime,
            double newNextArriveTime,
            double expectedArriveTime,
            double maxDelay,
            bool success )
```

Definition at line 28 of file InsertionResult.cs.

6.4.3 Property Documentation

6.4.3.1 ExpectedArriveTime

```
double VRPTWOptimizer.InsertionResult.ExpectedArriveTime  [get]
```

Time when we will arrive to serve inserted TransportRequest

Definition at line 11 of file InsertionResult.cs.

6.4.3.2 MaxDelay

```
double VRPTWOptimizer.InsertionResult.MaxDelay  [get]
```

Max delay generated by insertion of TransportRequest

Definition at line 15 of file InsertionResult.cs.

6.4.3.3 NewAddedDistance

```
double VRPTWOptimizer.InsertionResult.NewAddedDistance  [get]
```

Route length added by insertion of TransportRequest

Definition at line 19 of file InsertionResult.cs.

6.4.3.4 NewNextArriveTime

```
double VRPTWOptimizer.InsertionResult.NewNextArriveTime  [get]
```

Definition at line 20 of file InsertionResult.cs.

6.4.3.5 OldDistanceBetween

```
double VRPTWOptimizer.InsertionResult.OldDistanceBetween  [get]
```

Definition at line 22 of file InsertionResult.cs.

6.4.3.6 OldNextArriveTime

```
double VRPTWOptimizer.InsertionResult.OldNextArriveTime  [get]
```

Definition at line 24 of file InsertionResult.cs.

6.4.3.7 Success

```
bool VRPTWOptimizer.InsertionResult.Success  [get]
```

Definition at line 26 of file InsertionResult.cs.

The documentation for this class was generated from the following file:

- InsertionResult.cs

6.5 VRPTWOptimizer.Interfaces.IRoute Interface Reference

Represents route assigned to Vehicle

Properties

- List< double > ArrivalTimes [get]

  Array of arrival times to subsequent Location objects
- List< double > DepartureTimes [get]

  Array of departure times from subsequent Location objects
- List< Distance > Distances [get]

  Distance objects between subsequent location in route
- double Length [get]

  Total length of route in meters
- List< List< TransportRequest > > LoadedRequests [get]

  TransportRequest objects that are loaded on given location
- double MaxDelay [get]

  Timespan in seconds of largest delay againts PreferedTimeWindowEnd
- List< double > TimeWindowEnd [get]

  Array of upper time limits for starting visits in subsequent Location objects
- List< double > TimeWindowStart [get]

  Array of lower time limits for starting visits in subsequent Location objects
- double TotalDelay [get]

  Sum of all delays againts PreferedTimeWindowEnd
- double TravelTime [get]

  Total duration of drive within route in seconds
- List< List< TransportRequest > > UnloadedRequests [get]

  TransportRequest objects that are unloaded on given location
- Vehicle Vehicle [get]

  Vehicle with capacity serving the route
- Driver VehicleDriver [get]

  Tractor serving the route (if necessary)
- Vehicle VehicleTractor [get]

  Driver (if the problem assumes drivers management)
- List< Location > VisitedLocations [get]

  List of subsequent Location objects in route

6.5.1 Detailed Description

Represents route assigned to Vehicle

Definition at line 9 of file IRoute.cs.

6.5.2 Property Documentation

6.5.2.1 ArrivalTimes

```
List<double> VRPTWOptimizer.Interfaces.IRoute.ArrivalTimes [get]
```

Array of arrival times to subsequent Location objects

Definition at line 14 of file IRoute.cs.

6.5.2.2 DepartureTimes

```
List<double> VRPTWOptimizer.Interfaces.IRoute.DepartureTimes [get]
```

Array of departure times from subsequent Location objects

Definition at line 18 of file IRoute.cs.

6.5.2.3 Distances

```
List<Distance> VRPTWOptimizer.Interfaces.IRoute.Distances [get]
```

Distance objects between subsequent location in route

Definition at line 22 of file IRoute.cs.

6.5.2.4 Length

```
double VRPTWOptimizer.Interfaces.IRoute.Length [get]
```

Total length of route in meters

Definition at line 26 of file IRoute.cs.

6.5.2.5 LoadedRequests

```
List<List<TransportRequest> > VRPTWOptimizer.Interfaces.IRoute.LoadedRequests [get]
```

TransportRequest objects that are loaded on given location

Definition at line 30 of file IRoute.cs.

6.5.2.6 MaxDelay

```
double VRPTWOptimizer.Interfaces.IRoute.MaxDelay [get]
```

Timespan in seconds of largest delay againts PreferedTimeWindowEnd

Definition at line 34 of file IRoute.cs.

6.5.2.7 TimeWindowEnd

```
List<double> VRPTWOptimizer.Interfaces.IRoute.TimeWindowEnd [get]
```

Array of upper time limits for starting visits in subsequent Location objects

Definition at line 38 of file IRoute.cs.

6.5.2.8 TimeWindowStart

```
List<double> VRPTWOptimizer.Interfaces.IRoute.TimeWindowStart [get]
```

Array of lower time limits for starting visits in subsequent Location objects

Definition at line 42 of file IRoute.cs.

6.5.2.9 TotalDelay

```
double VRPTWOptimizer.Interfaces.IRoute.TotalDelay [get]
```

Sum of all delays againts PreferedTimeWindowEnd

Definition at line 46 of file IRoute.cs.

6.5.2.10 TravelTime

```
double VRPTWOptimizer.Interfaces.IRoute.TravelTime [get]
```

Total duration of drive within route in seconds

Definition at line 50 of file IRoute.cs.

6.5.2.11 UnloadedRequests

```
List<List<TransportRequest> > VRPTWOptimizer.Interfaces.IRoute.UnloadedRequests [get]
```

TransportRequest objects that are unloaded on given location

Definition at line 54 of file IRoute.cs.

6.5.2.12 Vehicle

```
Vehicle VRPTWOptimizer.Interfaces.IRoute.Vehicle [get]
```

Vehicle with capacity serving the route

Definition at line 58 of file IRoute.cs.

6.5.2.13 VehicleDriver

```
Driver VRPTWOptimizer.Interfaces.IRoute.VehicleDriver [get]
```

Tractor serving the route (if necessary)

Definition at line 62 of file IRoute.cs.

6.5.2.14 VehicleTractor

```
Vehicle VRPTWOptimizer.Interfaces.IRoute.VehicleTractor [get]
```

Driver (if the problem assumes drivers management)

Definition at line 66 of file IRoute.cs.

6.5.2.15 VisitedLocations

```
List<Location> VRPTWOptimizer.Interfaces.IRoute.VisitedLocations [get]
```

List of subsequent Location objects in route

Definition at line 70 of file IRoute.cs.

The documentation for this interface was generated from the following file:

- IRoute.cs

6.6 VRPTWOptimizer.Interfaces.ITimeEstimator Interface Reference

Predicts time of loading and unloading cargo at Location

Public Member Functions

- double EstimateLoadUnloadTime (int epUnloadCount, int epLoadOnlyCount, int epImmediatelyRetrieved↩ Count, int handledTransportRequestsCount)

  This method is used to predict time it will take to serve all the loadings and unloadings within the given location

6.6.1 Detailed Description

Predicts time of loading and unloading cargo at Location

Definition at line 6 of file ITimeEstimator.cs.

6.6.2 Member Function Documentation

6.6.2.1 EstimateLoadUnloadTime()

```
double VRPTWOptimizer.Interfaces.ITimeEstimator.EstimateLoadUnloadTime (
            int epUnloadCount,
            int epLoadOnlyCount,
            int epImmediatelyRetrievedCount,
            int handledTransportRequestsCount )
```

This method is used to predict time it will take to serve all the loadings and unloadings within the given location

Parameters

epUnloadCount	Number of EuroPallets that are being unloaded
epLoadOnlyCount	Number of EuroPallets that are being loaded, but were not unloaded
epImmediatelyRetrievedCount	Number of EuroPallets that are being delivered and immediately retrieved after handling in Location
handledTransportRequestsCount	Number of transport requests loaded and unloaded in the given location

Returns

The documentation for this interface was generated from the following file:

- ITimeEstimator.cs

6.7 VRPTWOptimizer.Interfaces.IVRPOptimizer Interface Reference

Optimizes Vehicle Routing Problem

Public Member Functions

- VRPOptimizerResult Optimize (IVRPProvider problemDataProvider, ITimeEstimator serviceTimeEstimator, IDistanceProvider distanceProvider)

  Solves Vehicle Routing Problem
- VRPOptimizerResult Optimize (IVRPProvider problemDataProvider, VRPCostFunction costFunctionFactors, ITimeEstimator serviceTimeEstimator, IDistanceProvider distanceProvider)

  Solves Vehicle Routing Problem

6.7.1 Detailed Description

Optimizes Vehicle Routing Problem

Definition at line 9 of file IVRPOptimizer.cs.

6.7.2 Member Function Documentation

6.7.2.1 Optimize() [1/2]

```
VRPOptimizerResult VRPTWOptimizer.Interfaces.IVRPOptimizer.Optimize (
            IVRPProvider problemDataProvider,
            ITimeEstimator serviceTimeEstimator,
            IDistanceProvider distanceProvider )
```

Solves Vehicle Routing Problem

Parameters

problemDataProvider	
serviceTimeEstimator	
distanceProvider	

Returns

6.7.2.2 Optimize() [2/2]

```
VRPOptimizerResult VRPTWOptimizer.Interfaces.IVRPOptimizer.Optimize (
            IVRPProvider problemDataProvider,
            VRPCostFunction costFunctionFactors,
            ITimeEstimator serviceTimeEstimator,
            IDistanceProvider distanceProvider )
```

Solves Vehicle Routing Problem

Parameters

problemDataProvider	
costFunctionFactors	
serviceTimeEstimator	
distanceProvider	

Returns

The documentation for this interface was generated from the following file:

- IVRPOptimizer.cs

6.8 VRPTWOptimizer.Interfaces.IVRPOptimizerFactory Interface Reference

Provides new instance of optimizer (follows Abstract Factory desing pattern)

Public Member Functions

- IVRPOptimizer CreateOptimizer ()
  Creates new instance of a VRP optimizer with default parameters
- IVRPOptimizer CreateOptimizer (Dictionary< string, object > configuration)
  Creates new instance of a VRP optimizer

6.8.1 Detailed Description

Provides new instance of optimizer (follows Abstract Factory desing pattern)

Definition at line 8 of file IVRPOptimizerFactory.cs.

6.8.2 Member Function Documentation

6.8.2.1 CreateOptimizer() [1/2]

```
IVRPOptimizer VRPTWOptimizer.Interfaces.IVRPOptimizerFactory.CreateOptimizer ( )
```

Creates new instance of a VRP optimizer with default parameters

Returns

6.8.2.2 CreateOptimizer() [2/2]

```
IVRPOptimizer VRPTWOptimizer.Interfaces.IVRPOptimizerFactory.CreateOptimizer (
            Dictionary< string, object > configuration )
```

Creates new instance of a VRP optimizer

Returns

The documentation for this interface was generated from the following file:

- IVRPOptimizerFactory.cs

6.9 VRPTWOptimizer.Interfaces.IVRPProvider Interface Reference

Provides problem data for IVRPOptimizer

Public Member Functions

- void LoadData (DateTime billingDate, string homeDepotId)
  Gets the data from underlying source

Properties

- List< Driver > Drivers [get]
  List of available Driver objects
- Location HomeDepot [get]
  Location of the main warehouse or main depot
- List< TransportRequest > Requests [get]
  List of TransportRequest objects to be served
- List< Vehicle > Vehicles [get]
  List of available Vehicle objects
- DateTime ZeroHour [get]
  Real world timestamp used to turn relative time of the problem to real world time

6.9.1 Detailed Description

Provides problem data for IVRPOptimizer

Definition at line 10 of file IVRPProvider.cs.

6.9.2 Member Function Documentation

6.9.2.1 LoadData()

```
void VRPTWOptimizer.Interfaces.IVRPProvider.LoadData (
            DateTime billingDate,
            string homeDepotId )
```

Gets the data from underlying source

Parameters

billingDate	
homeDepotId	

6.9.3 Property Documentation

6.9.3.1 Drivers

```
List<Driver> VRPTWOptimizer.Interfaces.IVRPProvider.Drivers [get]
```

List of available Driver objects

Definition at line 15 of file IVRPProvider.cs.

6.9.3.2 HomeDepot

```
Location VRPTWOptimizer.Interfaces.IVRPProvider.HomeDepot [get]
```

Location of the main warehouse or main depot

Definition at line 19 of file IVRPProvider.cs.

6.9.3.3 Requests

```
List<TransportRequest> VRPTWOptimizer.Interfaces.IVRPProvider.Requests [get]
```

List of TransportRequest objects to be served

Definition at line 23 of file IVRPProvider.cs.

6.9.3.4 Vehicles

```
List<Vehicle> VRPTWOptimizer.Interfaces.IVRPProvider.Vehicles [get]
```

List of available Vehicle objects

Definition at line 27 of file IVRPProvider.cs.

6.9.3.5 ZeroHour

```
DateTime VRPTWOptimizer.Interfaces.IVRPProvider.ZeroHour  [get]
```

Real world timestamp used to turn relative time of the problem to real world time

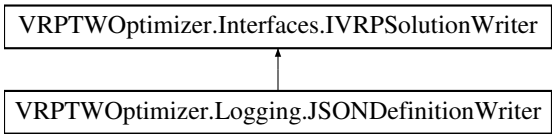
Definition at line 31 of file IVRPProvider.cs.

The documentation for this interface was generated from the following file:

- IVRPProvider.cs

6.10 VRPTWOptimizer.Interfaces.IVRPSolutionWriter Interface Reference

Inheritance diagram for VRPTWOptimizer.Interfaces.IVRPSolutionWriter:



Public Member Functions

- void SaveSolution (List< IRoute > routes, List< TransportRequest > unassignedRequests, DateTime billingDate, string homeDepotId, string algorithmName)

6.10.1 Detailed Description

Definition at line 6 of file IVRPSolutionWriter.cs.

6.10.2 Member Function Documentation

6.10.2.1 SaveSolution()

```
void VRPTWOptimizer.Interfaces.IVRPSolutionWriter.SaveSolution (
            List< IRoute > routes,
            List< TransportRequest > unassignedRequests,
            DateTime billingDate,
            string homeDepotId,
            string algorithmName )
```

Implemented in VRPTWOptimizer.Logging.JSONDefinitionWriter.

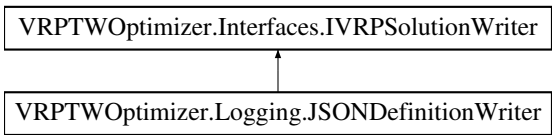
The documentation for this interface was generated from the following file:

- IVRPSolutionWriter.cs

6.11 VRPTWOptimizer.Logging.JSONDefinitionWriter Class Reference

Serializes solution to JSON

Inheritance diagram for VRPTWOptimizer.Logging.JSONDefinitionWriter:



Public Member Functions

- JSONDefinitionWriter (List< TransportRequest > requests, List< Vehicle > vehicles, IVRPOptimizer optimizer, DateTime zeroHour, DateTime computationsEnd, DateTime computationsStart, IDistanceProvider distanceData, ITimeEstimator timeEstimator, string filename, string clientName)
- void SaveSolution (List< IRoute > routes, List< TransportRequest > unassignedRequests, DateTime billingDate, string homeDepotId, string algorithmName)

  Saves solution to the format implemented by this Writer class (JSON in this case)

6.11.1 Detailed Description

Serializes solution to JSON

Definition at line 14 of file JSONDefinitionWriter.cs.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 JSONDefinitionWriter()

```
VRPTWOptimizer.Logging.JSONDefinitionWriter.JSONDefinitionWriter (
            List< TransportRequest > requests,
            List< Vehicle > vehicles,
            IVRPOptimizer optimizer,
            DateTime zeroHour,
            DateTime computationsEnd,
            DateTime computationsStart,
            IDistanceProvider distanceData,
            ITimeEstimator timeEstimator,
            string filename,
            string clientName )
```

Definition at line 29 of file JSONDefinitionWriter.cs.

6.11.3 Member Function Documentation

6.11.3.1 SaveSolution()

```
void VRPTWOptimizer.Logging.JSONDefinitionWriter.SaveSolution (
            List< IRoute > routes,
            List< TransportRequest > unassignedRequests,
            DateTime billingDate,
            string homeDepotId,
            string algorithmName )
```

Saves solution to the format implemented by this Writer class (JSON in this case)

Parameters

routes	
unassignedRequests	
billingDate	
homeDepotId	
algorithmName	

Implements VRPTWOptimizer.Interfaces.IVRPSolutionWriter.

Definition at line 60 of file JSONDefinitionWriter.cs.

The documentation for this class was generated from the following file:

- JSONDefinitionWriter.cs

6.12 VRPTWOptimizer.Dto.PickingSchedule Class Reference

Complete picking schedule for one day

Public Member Functions

- void TrySaveToFile (string filename)

  Writes picking schedule to designated JSON file

Static Public Member Functions

- static PickingSchedule GeneratePickingSchedule (VRPDefinition vrpDefinition, VRPSolution vrpSolution)

  Creates picking schedule from VRP definition and solution
- static PickingSchedule GeneratePickingSchedule (List< IRoute > routes, List< Vehicle > vehicles)

  Creates picking schedule from VRP routes and available vehicles

Properties

- string CallbackUrl [get, set]

  URI where updated picking schedule should be sent back
- int Id [get, set]

  Identifier of picking schedule
- DateTime OrdersCreateDate [get, set]

  DateTime when picking schedule was generated
- DateTime OrdersPickingStart [get, set]

  DateTime when first shift of picking starts
- List< TransportPickingLists > PickingLists [get, set]

  Picking orders divided among transports
- List< VehicleSchedule > VehiclesAvailability [get, set]

  Schedule of vehicles presence within the warehouse bounds

6.12.1 Detailed Description

Complete picking schedule for one day

Definition at line 15 of file PickingSchedule.cs.

6.12.2 Member Function Documentation

6.12.2.1 GeneratePickingSchedule() [1/2]

```
static PickingSchedule VRPTWOptimizer.Dto.PickingSchedule.GeneratePickingSchedule (
            List< IRoute > routes,
            List< Vehicle > vehicles )  [static]
```

Creates picking schedule from VRP routes and available vehicles

Parameters

routes	
vehicles	

Returns

Definition at line 149 of file PickingSchedule.cs.

6.12.2.2 GeneratePickingSchedule() [2/2]

```
static PickingSchedule VRPTWOptimizer.Dto.PickingSchedule.GeneratePickingSchedule (
            VRPDefinition vrpDefinition,
            VRPSolution vrpSolution )  [static]
```

Creates picking schedule from VRP definition and solution

Parameters

vrpDefinition	
vrpSolution	

Returns

Definition at line 55 of file PickingSchedule.cs.

6.12.2.3 TrySaveToFile()

```
void VRPTWOptimizer.Dto.PickingSchedule.TrySaveToFile (
            string filename )
```

Writes picking schedule to designated JSON file

Parameters

filename	

Definition at line 158 of file PickingSchedule.cs.

6.12.3 Property Documentation

6.12.3.1 CallbackUrl

```
string VRPTWOptimizer.Dto.PickingSchedule.CallbackUrl  [get], [set]
```

URI where updated picking schedule should be sent back

Definition at line 22 of file PickingSchedule.cs.

6.12.3.2 Id

```
int VRPTWOptimizer.Dto.PickingSchedule.Id  [get], [set]
```

Identifier of picking schedule

Definition at line 27 of file PickingSchedule.cs.

6.12.3.3 OrdersCreateDate

```
DateTime VRPTWOptimizer.Dto.PickingSchedule.OrdersCreateDate  [get], [set]
```

DateTime when picking schedule was generated

Definition at line 32 of file PickingSchedule.cs.

6.12.3.4 OrdersPickingStart

```
DateTime VRPTWOptimizer.Dto.PickingSchedule.OrdersPickingStart  [get], [set]
```

DateTime when first shift of picking starts

Definition at line 37 of file PickingSchedule.cs.

6.12.3.5 PickingLists

```
List<TransportPickingLists> VRPTWOptimizer.Dto.PickingSchedule.PickingLists  [get], [set]
```

Picking orders divided among transports

Definition at line 42 of file PickingSchedule.cs.

6.12.3.6 VehiclesAvailability

```
List<VehicleSchedule> VRPTWOptimizer.Dto.PickingSchedule.VehiclesAvailability  [get], [set]
```

Schedule of vehicles presence within the warehouse bounds

Definition at line 47 of file PickingSchedule.cs.

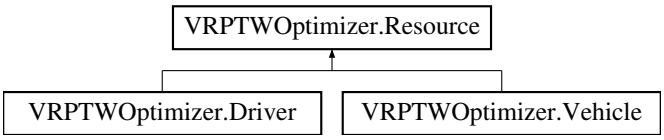
The documentation for this class was generated from the following file:

- PickingSchedule.cs

6.13 VRPTWOptimizer.Resource Class Reference

Generalized time bound resource (driver, machine, vehicle)

Inheritance diagram for VRPTWOptimizer.Resource:



Public Member Functions

- Resource (int id, double availabilityStart, double availabilityEnd)

  Creates generic Resource object

Properties

- double AvailabilityEnd `[get, protected set]`

  Upper bound of Resource time availability (suggestion)
- double AvailabilityStart `[get, protected set]`

  Lower bound of Resource time availability (strict)
- int Id `[get, protected set]`

  Identifier of the Resource

6.13.1 Detailed Description

Generalized time bound resource (driver, machine, vehicle)

Definition at line 8 of file Resource.cs.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 Resource()

```
VRPTWOptimizer.Resource.Resource (
            int id,
            double availabilityStart,
            double availabilityEnd )
```

Creates generic Resource object

Parameters

id	
availabilityStart	
availabilityEnd	

Definition at line 32 of file Resource.cs.

6.13.3 Property Documentation

6.13.3.1 AvailabilityEnd

```
double VRPTWOptimizer.Resource.AvailabilityEnd  [get], [protected set]
```

Upper bound of Resource time availability (suggestion)

Definition at line 14 of file Resource.cs.

6.13.3.2 AvailabilityStart

```
double VRPTWOptimizer.Resource.AvailabilityStart  [get], [protected set]
```

Lower bound of Resource time availability (strict)

Definition at line 19 of file Resource.cs.

6.13.3.3 Id

```
int VRPTWOptimizer.Resource.Id  [get], [protected set]
```

Identifier of the Resource

Definition at line 24 of file Resource.cs.

The documentation for this class was generated from the following file:

- Resource.cs

6.14 VRPTWOptimizer.VRPSolution.ScheduleItem Class Reference

Single time entry describing planned visit at a given Location

Properties

- double ArrivalTime [get, set]

  Relative time in seconds when Vehicle arrives at Location
- double Delay [get, set]

  Delay in seconds againt the TimeWindowEnd if specified
- double DepartureTime [get, set]

  Relative time in seconds when Vehicle leaves the Location
- List< int > LoadedRequestsIds [get, set]

  Identifiers of TransportRequest objects that are loaded onto Vehicle at Location
- string LocationId [get, set]

  Identifier of visited Location
- List< int > UnloadedRequestsIds [get, set]

  Identifiers of TransportRequest objects that are unloaded from Vehicle at Location

6.14.1 Detailed Description

Single time entry describing planned visit at a given Location

Definition at line 18 of file VRPSolution.cs.

6.14.2 Property Documentation

6.14.2.1 ArrivalTime

```
double VRPTWOptimizer.VRPSolution.ScheduleItem.ArrivalTime [get], [set]
```

Relative time in seconds when Vehicle arrives at Location

Definition at line 23 of file VRPSolution.cs.

6.14.2.2 Delay

```
double VRPTWOptimizer.VRPSolution.ScheduleItem.Delay [get], [set]
```

Delay in seconds againt the TimeWindowEnd if specified

Definition at line 27 of file VRPSolution.cs.

6.14.2.3 DepartureTime

```
double VRPTWOptimizer.VRPSolution.ScheduleItem.DepartureTime  [get], [set]
```

Relative time in seconds when Vehicle leaves the Location

Definition at line 31 of file VRPSolution.cs.

6.14.2.4 LoadedRequestsIds

```
List<int> VRPTWOptimizer.VRPSolution.ScheduleItem.LoadedRequestsIds  [get], [set]
```

Identifiers of TransportRequest objects that are loaded onto Vehicle at Location

Definition at line 35 of file VRPSolution.cs.

6.14.2.5 LocationId

```
string VRPTWOptimizer.VRPSolution.ScheduleItem.LocationId  [get], [set]
```

Identifier of visited Location

Definition at line 39 of file VRPSolution.cs.

6.14.2.6 UnloadedRequestsIds

```
List<int> VRPTWOptimizer.VRPSolution.ScheduleItem.UnloadedRequestsIds  [get], [set]
```

Identifiers of TransportRequest objects that are unloaded from Vehicle at Location

Definition at line 44 of file VRPSolution.cs.

The documentation for this class was generated from the following file:

- VRPSolution.cs

6.15 VRPTWOptimizer.Dto.StorePickingList Class Reference

Picking order details for single store

Properties

- string DeliveryLocationId [get, set]
  Store identifier
- double EpCount [get, set]
  Total cargo size in europallets
- List< CargoUnit > GoodsList [get, set]
  List of ordered goods
- int LoadingOrder [get, set]
  Loading order 1st to load is the last to deliver (stores form a stack of deliveries within the truck cargo hold)

6.15.1 Detailed Description

Picking order details for single store

Definition at line 13 of file StorePickingList.cs.

6.15.2 Property Documentation

6.15.2.1 DeliveryLocationId

```
string VRPTWOptimizer.Dto.StorePickingList.DeliveryLocationId [get], [set]
```

Store identifier

Definition at line 19 of file StorePickingList.cs.

6.15.2.2 EpCount

```
double VRPTWOptimizer.Dto.StorePickingList.EpCount [get], [set]
```

Total cargo size in europallets

Definition at line 24 of file StorePickingList.cs.

6.15.2.3 GoodsList

```
List<CargoUnit> VRPTWOptimizer.Dto.StorePickingList.GoodsList [get], [set]
```

List of ordered goods

Definition at line 29 of file StorePickingList.cs.

6.15.2.4 LoadingOrder

```
int VRPTWOptimizer.Dto.StorePickingList.LoadingOrder [get], [set]
```

Loading order 1st to load is the last to deliver (stores form a stack of deliveries within the truck cargo hold)

Definition at line 34 of file StorePickingList.cs.

The documentation for this class was generated from the following file:

- StorePickingList.cs

6.16 VRPTWOptimizer.Dto.TimeInterval Class Reference

Represents time interval for vehicle yard availability

Properties

- DateTime AvailabilityEnd `[get, set]`
  End of the interval (vehicle leaves warehouse grounds)
- DateTime AvailabilityStart `[get, set]`
  Start of the interval (vehicle within warehouse grounds)

6.16.1 Detailed Description

Represents time interval for vehicle yard availability

Definition at line 13 of file TimeInterval.cs.

6.16.2 Property Documentation

6.16.2.1 AvailabilityEnd

```
DateTime VRPTWOptimizer.Dto.TimeInterval.AvailabilityEnd [get], [set]
```

End of the interval (vehicle leaves warehouse grounds)

Definition at line 19 of file TimeInterval.cs.

6.16.2.2 AvailabilityStart

```
DateTime VRPTWOptimizer.Dto.TimeInterval.AvailabilityStart [get], [set]
```

Start of the interval (vehicle within warehouse grounds)

Definition at line 24 of file TimeInterval.cs.

The documentation for this class was generated from the following file:

- TimeInterval.cs

6.17 VRPTWOptimizer.VRPSolution.TransportItem Class Reference

Entry describing a single loop of the Vehicle/combined Vehicle

Properties

- double AvailableForLoadingTime `[get, set]`
  Relative time when the vehicle needs to be at the gate of warehouse to be loaded
- double AvailableForNextAssignmentTime `[get, set]`
  Relative time when the vehicle is free for next assignements
- int DriverId `[get, set]`
- double FillInRatio `[get, set]`
  Percent of capacity filled when starting the Route
- double Length `[get, set]`
  Length of a routes in meters
- List< ScheduleItem > Schedule `[get, set]`
  List of planned visits
- int TractorId `[get, set]`
  Identifier of tractor unit (if applicable)
- int TrailerTruckId `[get, set]`
  Identifier of semitrailer or straight truck
- int TransportId `[get, set]`
  Route identifier

6.17.1 Detailed Description

Entry describing a single loop of the Vehicle/combined Vehicle

Definition at line 50 of file VRPSolution.cs.

6.17.2 Property Documentation

6.17.2.1 AvailableForLoadingTime

```
double VRPTWOptimizer.VRPSolution.TransportItem.AvailableForLoadingTime  [get], [set]
```

Relative time when the vehicle needs to be at the gate of warehouse to be loaded

Definition at line 55 of file VRPSolution.cs.

6.17.2.2 AvailableForNextAssignmentTime

```
double VRPTWOptimizer.VRPSolution.TransportItem.AvailableForNextAssignmentTime  [get], [set]
```

Relative time when the vehicle is free for next assignements

Definition at line 59 of file VRPSolution.cs.

6.17.2.3 DriverId

```
int VRPTWOptimizer.VRPSolution.TransportItem.DriverId  [get], [set]
```

Identifier of Driver (if applicable)

Definition at line 64 of file VRPSolution.cs.

6.17.2.4 FillInRatio

```
double VRPTWOptimizer.VRPSolution.TransportItem.FillInRatio  [get], [set]
```

Percent of capacity filled when starting the Route

Definition at line 68 of file VRPSolution.cs.

6.17.2.5 Length

```
double VRPTWOptimizer.VRPSolution.TransportItem.Length  [get], [set]
```

Length of a routes in meters

Definition at line 72 of file VRPSolution.cs.

6.17.2.6 Schedule

```
List<ScheduleItem> VRPTWOptimizer.VRPSolution.TransportItem.Schedule [get], [set]
```

List of planned visits

Definition at line 76 of file VRPSolution.cs.

6.17.2.7 TractorId

```
int VRPTWOptimizer.VRPSolution.TransportItem.TractorId [get], [set]
```

Identifier of tractor unit (if applicable)

Definition at line 80 of file VRPSolution.cs.

6.17.2.8 TrailerTruckId

```
int VRPTWOptimizer.VRPSolution.TransportItem.TrailerTruckId [get], [set]
```

Identifier of semitrailer or straight truck

Definition at line 84 of file VRPSolution.cs.

6.17.2.9 TransportId

```
int VRPTWOptimizer.VRPSolution.TransportItem.TransportId [get], [set]
```

Route identifier

Definition at line 88 of file VRPSolution.cs.

The documentation for this class was generated from the following file:

- VRPSolution.cs

6.18 VRPTWOptimizer.Dto.TransportPickingLists Class Reference

Picking orders for single transport

Properties

- VehicleType CapacityVehicleType `[get, set]`

  Type of capacity vehicle: integrated truck or semi-trailer
- DateTime DesiredDepartureTime `[get, set]`

  DateTime when the truck is designed to leave the warehouse
- int EpCapacity `[get, set]`

  Size of selected vehicle in europallets
- int MaxEpCapacity `[get, set]`

  Maximum allowed sized for the truck - minimum of stores upper limits
- int SemiTrailerTruckId `[get, set]`

  Identifier of truck or semi-trailer which will peroform the transport
- List< StorePickingList > StoreOrders `[get, set]`

  List of all goods requested by store
- int TransportId `[get, set]`

  Identifier of transport (single route/loop)

6.18.1 Detailed Description

Picking orders for single transport

Definition at line 14 of file TransportPickingLists.cs.

6.18.2 Property Documentation

6.18.2.1 CapacityVehicleType

```
VehicleType VRPTWOptimizer.Dto.TransportPickingLists.CapacityVehicleType [get], [set]
```

Type of capacity vehicle: integrated truck or semi-trailer

Definition at line 20 of file TransportPickingLists.cs.

6.18.2.2 DesiredDepartureTime

```
DateTime VRPTWOptimizer.Dto.TransportPickingLists.DesiredDepartureTime [get], [set]
```

DateTime when the truck is designed to leave the warehouse

Definition at line 26 of file TransportPickingLists.cs.

6.18.2.3 EpCapacity

```
int VRPTWOptimizer.Dto.TransportPickingLists.EpCapacity [get], [set]
```

Size of selected vehicle in europallets

Definition at line 31 of file TransportPickingLists.cs.

6.18.2.4 MaxEpCapacity

```
int VRPTWOptimizer.Dto.TransportPickingLists.MaxEpCapacity [get], [set]
```

Maximum allowed sized for the truck - minimum of stores upper limits

Definition at line 36 of file TransportPickingLists.cs.

6.18.2.5 SemiTrailerTruckId

```
int VRPTWOptimizer.Dto.TransportPickingLists.SemiTrailerTruckId [get], [set]
```

Identifier of truck or semi-trailer which will peroform the transport

Definition at line 41 of file TransportPickingLists.cs.

6.18.2.6 StoreOrders

```
List<StorePickingList> VRPTWOptimizer.Dto.TransportPickingLists.StoreOrders [get], [set]
```

List of all goods requested by store

Definition at line 46 of file TransportPickingLists.cs.

6.18.2.7 TransportId

```
int VRPTWOptimizer.Dto.TransportPickingLists.TransportId [get], [set]
```

Identifier of transport (single route/loop)

Definition at line 51 of file TransportPickingLists.cs.

The documentation for this class was generated from the following file:

- TransportPickingLists.cs

6.19 VRPTWOptimizer.TransportRequest Class Reference

Description of the request to move cargo from pickup to delivery Location

Public Member Functions

- TransportRequest (int id, double[] size, int[] necessaryVehicleSpecialProperties, int packageCount, int packageCountForImediateRetrieval, Location startLocation, double pickupAvailableTimeWindowStart, double pickupPreferedTimeWindowStart, double pickupPreferedTimeWindowEnd, double pickupAvailableTimeWindowEnd, Location endLocation, double deliveryAvailableTimeWindowStart, double deliveryPreferedTimeWindowStart, double deliveryPreferedTimeWindowEnd, double deliveryAvailableTimeWindowEnd, RequestType requestType, int[] cargoTypes, VehicleRoadRestrictionProperties maxVehicleSize, int[] restrictedGoodsTypes, Dictionary< int, double > mutuallyExclusiveRequestsIdTimeBufferDict, double revenueValue, string name)

  Creates generic TransportRequest when class is inherited from it will have to use this to define problem
- List< TransportRequest > ExtractBestFitRequests (List< TransportRequest > requests, ITimeEstimator timeEstimator, IDistanceProvider distanceProvider)

  Find requests that should be paired with this TransportRequest

Properties

- int CargoType [get]

  Main cargo type
- int[] CargoTypes [get, protected set]

  All cargo types within request
- double DeliveryAvailableTimeWindowEnd [get, protected set]

  Time window end while request can still be delivered
- double DeliveryAvailableTimeWindowStart [get, protected set]

  Earlies time window start when request can be delivered
- Location DeliveryLocation [get, protected set]

  Cargo destination
- double DeliveryPreferedTimeWindowEnd [get, protected set]

  Delivery time window end
- double DeliveryPreferedTimeWindowStart [get, protected set]

  Delivery time window start
- int Id [get, protected set]

  Numeric request identifier
- VehicleRoadRestrictionProperties MaxVehicleSize [get, protected set]

  Upper bound restrictions of vehicle parameters
- Dictionary< int, double > MutuallyExclusiveRequestsIdTimeBufferDict [get, protected set]

  Dictionary of requests that can be visited only after certain time of departure from end location or must be served certain time before arrival to end location
- string Name [get, protected set]

  Custom request identifier
- int[] NecessaryVehicleSpecialProperties [get, protected set]

  List of domain specific features that the vehicle needs to have to serve this request (e.g. freezer, lift) vector needs to conform to Vehicle.SpecialProperties
- int PackageCount [get, protected set]

  Total count of items to be delivered (e.g. Euro Pallet and DHP pallet are both a single item)
- int PackageCountForImediateRetrieval [get, protected set]

Subgroup of PackageCount property which are to be retrieved just after delivery

- double PickupAvailableTimeWindowEnd `[get, protected set]`

  Time window end while request can still be picked up
- double PickupAvailableTimeWindowStart `[get, protected set]`

  Earlies time window start when request can be picked up
- Location PickupLocation `[get, protected set]`

  Location where the cargo is picked up
- double PickupPreferedTimeWindowEnd `[get, protected set]`

  Time window end when request should be picked up
- double PickupPreferedTimeWindowStart `[get, protected set]`

  Time window start when request should be picked up
- int[ ] RestrictedCargoTypes `[get, protected set]`

  Cargo types identifiers that could not be transported together with this request
- double RevenueValue `[get, protected set]`

  Additional value gained if request is completed
- double[ ] Size `[get, protected set]`

  Total cargo size (Euro pallets count, mass, cubic meters etc.) Must conform to Vehicle capacity definition
- RequestType Type `[get, protected set]`

  Type of request: distribution, backhauling etc.

6.19.1 Detailed Description

Description of the request to move cargo from pickup to delivery Location

Definition at line 15 of file TransportRequest.cs.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 TransportRequest()

```
VRPTWOptimizer.TransportRequest.TransportRequest (
            int id,
            double[] size,
            int[] necessaryVehicleSpecialProperties,
            int packageCount,
            int packageCountForImediateRetrieval,
            Location startLocation,
            double pickupAvailableTimeWindowStart,
            double pickupPreferedTimeWindowStart,
            double pickupPreferedTimeWindowEnd,
            double pickupAvailableTimeWindowEnd,
            Location endLocation,
            double deliveryAvailableTimeWindowStart,
            double deliveryPreferedTimeWindowStart,
            double deliveryPreferedTimeWindowEnd,
            double deliveryAvailableTimeWindowEnd,
            RequestType requestType,
            int[] cargoTypes,
```

```
            VehicleRoadRestrictionProperties maxVehicleSize,
            int[] restrictedGoodsTypes,
            Dictionary< int, double > mutuallyExclusiveRequestsIdTimeBufferDict,
            double revenueValue,
            string name )
```

Creates generic TransportRequest when class is inherited from it will have to use this to define problem

Parameters

id	
size	
necessaryVehicleSpecialProperties	
packageCount	
packageCountForImediateRetrieval	
startLocation	
pickupAvailableTimeWindowStart	
pickupPreferedTimeWindowStart	
pickupPreferedTimeWindowEnd	
pickupAvailableTimeWindowEnd	
endLocation	
deliveryAvailableTimeWindowStart	
deliveryPreferedTimeWindowStart	
deliveryPreferedTimeWindowEnd	
deliveryAvailableTimeWindowEnd	
requestType	
cargoTypes	
maxVehicleSize	
restrictedGoodsTypes	
mutuallyExclusiveRequestsIdTimeBufferDict	
revenueValue	
name	

Definition at line 150 of file TransportRequest.cs.

6.19.3 Member Function Documentation

6.19.3.1 ExtractBestFitRequests()

```
List< TransportRequest > VRPTWOptimizer.TransportRequest.ExtractBestFitRequests (
            List< TransportRequest > requests,
            ITimeEstimator timeEstimator,
            IDistanceProvider distanceProvider )
```

Find requests that should be paired with this TransportRequest

Parameters

requests	
timeEstimator	
distanceProvider	

Returns

Definition at line 213 of file TransportRequest.cs.

6.19.4 Property Documentation

6.19.4.1 CargoType

```
int VRPTWOptimizer.TransportRequest.CargoType  [get]
```

Main cargo type

Definition at line 20 of file TransportRequest.cs.

6.19.4.2 CargoTypes

```
int [] VRPTWOptimizer.TransportRequest.CargoTypes  [get], [protected set]
```

All cargo types within request

Definition at line 24 of file TransportRequest.cs.

6.19.4.3 DeliveryAvailableTimeWindowEnd

```
double VRPTWOptimizer.TransportRequest.DeliveryAvailableTimeWindowEnd  [get], [protected set]
```

Time window end while request can still be delivered

Definition at line 28 of file TransportRequest.cs.

6.19.4.4 DeliveryAvailableTimeWindowStart

```
double VRPTWOptimizer.TransportRequest.DeliveryAvailableTimeWindowStart  [get], [protected set]
```

Earlies time window start when request can be delivered

Definition at line 32 of file TransportRequest.cs.

6.19.4.5 DeliveryLocation

```
Location VRPTWOptimizer.TransportRequest.DeliveryLocation  [get], [protected set]
```

Cargo destination

Definition at line 36 of file TransportRequest.cs.

6.19.4.6 DeliveryPreferedTimeWindowEnd

```
double VRPTWOptimizer.TransportRequest.DeliveryPreferedTimeWindowEnd  [get], [protected set]
```

Delivery time window end

Definition at line 41 of file TransportRequest.cs.

6.19.4.7 DeliveryPreferedTimeWindowStart

```
double VRPTWOptimizer.TransportRequest.DeliveryPreferedTimeWindowStart  [get], [protected set]
```

Delivery time window start

Definition at line 46 of file TransportRequest.cs.

6.19.4.8 Id

```
int VRPTWOptimizer.TransportRequest.Id  [get], [protected set]
```

Numeric request identifier

Definition at line 50 of file TransportRequest.cs.

6.19.4.9 MaxVehicleSize

```
VehicleRoadRestrictionProperties VRPTWOptimizer.TransportRequest.MaxVehicleSize [get], [protected
set]
```

Upper bound restrictions of vehicle parameters

Definition at line 54 of file TransportRequest.cs.

6.19.4.10 MutuallyExclusiveRequestsIdTimeBufferDict

```
Dictionary<int, double> VRPTWOptimizer.TransportRequest.MutuallyExclusiveRequestsIdTime↩
BufferDict [get], [protected set]
```

Dictionary of requests that can be visited only after certain time of departure from end location or must be served certain time before arrival to end location

Definition at line 59 of file TransportRequest.cs.

6.19.4.11 Name

```
string VRPTWOptimizer.TransportRequest.Name [get], [protected set]
```

Custom request identifier

Definition at line 63 of file TransportRequest.cs.

6.19.4.12 NecessaryVehicleSpecialProperties

```
int [] VRPTWOptimizer.TransportRequest.NecessaryVehicleSpecialProperties [get], [protected
set]
```

List of domain specific features that the vehicle needs to have to serve this request (e.g. freezer, lift) vector needs to conform to Vehicle.SpecialProperties

Definition at line 68 of file TransportRequest.cs.

6.19.4.13 PackageCount

```
int VRPTWOptimizer.TransportRequest.PackageCount [get], [protected set]
```

Total count of items to be delivered (e.g. Euro Pallet and DHP pallet are both a single item)

Definition at line 73 of file TransportRequest.cs.

6.19.4.14 PackageCountForImediateRetrieval

```
int VRPTWOptimizer.TransportRequest.PackageCountForImediateRetrieval  [get], [protected set]
```

Subgroup of PackageCount property which are to be retrieved just after delivery

Definition at line 78 of file TransportRequest.cs.

6.19.4.15 PickupAvailableTimeWindowEnd

```
double VRPTWOptimizer.TransportRequest.PickupAvailableTimeWindowEnd  [get], [protected set]
```

Time window end while request can still be picked up

Definition at line 83 of file TransportRequest.cs.

6.19.4.16 PickupAvailableTimeWindowStart

```
double VRPTWOptimizer.TransportRequest.PickupAvailableTimeWindowStart  [get], [protected set]
```

Earlies time window start when request can be picked up

Definition at line 88 of file TransportRequest.cs.

6.19.4.17 PickupLocation

```
Location VRPTWOptimizer.TransportRequest.PickupLocation  [get], [protected set]
```

Location where the cargo is picked up

Definition at line 93 of file TransportRequest.cs.

6.19.4.18 PickupPreferedTimeWindowEnd

```
double VRPTWOptimizer.TransportRequest.PickupPreferedTimeWindowEnd  [get], [protected set]
```

Time window end when request should be picked up

Definition at line 98 of file TransportRequest.cs.

6.19.4.19 PickupPreferedTimeWindowStart

```
double VRPTWOptimizer.TransportRequest.PickupPreferedTimeWindowStart  [get], [protected set]
```

Time window start when request should be picked up

Definition at line 103 of file TransportRequest.cs.

6.19.4.20 RestrictedCargoTypes

```
int [] VRPTWOptimizer.TransportRequest.RestrictedCargoTypes  [get], [protected set]
```

Cargo types identifiers that could not be transported together with this request

Definition at line 108 of file TransportRequest.cs.

6.19.4.21 RevenueValue

```
double VRPTWOptimizer.TransportRequest.RevenueValue  [get], [protected set]
```

Additional value gained if request is completed

Definition at line 113 of file TransportRequest.cs.

6.19.4.22 Size

```
double [] VRPTWOptimizer.TransportRequest.Size  [get], [protected set]
```

Total cargo size (Euro pallets count, mass, cubic meters etc.) Must conform to Vehicle capacity definition

Definition at line 119 of file TransportRequest.cs.

6.19.4.23 Type

```
RequestType VRPTWOptimizer.TransportRequest.Type  [get], [protected set]
```

Type of request: distribution, backhauling etc.

Definition at line 123 of file TransportRequest.cs.

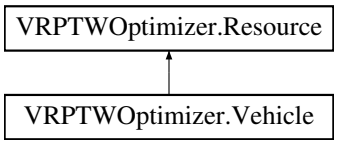
The documentation for this class was generated from the following file:

- TransportRequest.cs

6.20 VRPTWOptimizer.Vehicle Class Reference

Defines properties of a Vehicle

Inheritance diagram for VRPTWOptimizer.Vehicle:



Public Member Functions

- Vehicle (int id, double[] capacity, int[] specialProperties, Aggregation[] capacityAggregation, Location initialLocation, double availabilityStart, Location finalLocation, double availabilityEnd, double maxRide↩ Time, VehicleRoadRestrictionProperties roadProperties, VehicleType type, double vehicleCostPerDistance↩ Unit, double vehicleCostPerTimeUnit, double vehicleCostPerUsage, int ownerID)

  Creates vehicle object for backward compatibility
- Vehicle (int id, double[] capacity, int[] specialProperties, Aggregation[] capacityAggregation, Location initialLocation, double availabilityStart, Location finalLocation, double availabilityEnd, double maxRide↩ Time, VehicleRoadRestrictionProperties roadProperties, VehicleType type, double vehicleCostPerDistance↩ Unit, double vehicleCostPerTimeUnit, double vehicleCostPerUsage, int ownerID, double vehicleFlatCost↩ ForShortRouteLength, double vehicleMaxRouteLengthForFlatCost)

  Creates vehicle object
- bool CanFitRequests (IEnumerable< TransportRequest > requests)

  Verifies if a pool of TransportRequest objects can fit together within the Vehicle
- bool CanFitRequestsSomewhereInVehicle (IEnumerable< TransportRequest > requests)

  Verifies if a pool of TransportRequest objects can fit together within the Vehicle
- bool CanHandleRequest (TransportRequest candidateRequest)

  Verifies if properties of TransportRequest conform with abilities of Vehicle and restrictions of TransportRequest allow the Vehicle to approach initial and final destinations

Static Public Member Functions

- static bool CanFitCapacity (double[] capacity, Aggregation[] aggregationType, IEnumerable< double[]> sizes)

  Checks if the given list of cargo sizes would fit into specified capacity according to proper size aggregation

Properties

- double[] Capacity `[get, protected set]`

  Array of capacity dimensions (e.g. mass, length, volume, euro pallets count) Freezer availability can also be treated as dimension
- Aggregation[] CapacityAggregationType `[get, protected set]`

  Array of how the packages sizes are aggregated in various dimensions (e.g. sum for mass, max for length)
- Location FinalLocation `[get, protected set]`

  Location where vehicle needs to finish its operations
- Location InitialLocation `[get, protected set]`

  Location where vehicle will be available at AvailabilityStart

- double MaxRideTime [get, protected set]

  Max time span when vehicle can be on road
- int OwnerID [get, protected set]

  Identifier of the company owning the vehicle
- VehicleRoadRestrictionProperties RoadProperties [get, protected set]

  Size of the vehicle that affects possibility of traversing given route or accessing a certain Location
- int[] SpecialProperties [get, protected set]

  Describes properties of vehicles within a given domain (e.g. lift, freezer)
- VehicleType Type [get, protected set]

  Type of vehicle necessary to decide if needs to be combined with a unit with engine (semi-trailer) or not (truck)
- double VehicleCostPerDistanceUnit [get, protected set]

  Cost of using vehicle per distance unit
- double VehicleCostPerTimeUnit [get, protected set]

  Cost of using vehicle per time unit (while moving from Location to Location)
- double VehicleCostPerUsage [get, protected set]

  Cost of using vehicle at all in a given problem
- double VehicleFlatCostForShortRouteLength [get]

  Cost of using vehicle if route length is shorter than VehicleMaxRouteLengthForFlatCost (in meters)
- double VehicleMaxRouteLengthForFlatCost [get]

  Distance (by default in meters) for which VehicleFlatCostForShortRouteLength flat rate is applied, otherwise Vehicle↩ CostPerDistanceUnit ∗ Distance is applieds

6.20.1 Detailed Description

Defines properties of a Vehicle

Definition at line 13 of file Vehicle.cs.

6.20.2 Constructor & Destructor Documentation

6.20.2.1 Vehicle() [1/2]

```
VRPTWOptimizer.Vehicle.Vehicle (
            int id,
            double[] capacity,
            int[] specialProperties,
            Aggregation[] capacityAggregation,
            Location initialLocation,
            double availabilityStart,
            Location finalLocation,
            double availabilityEnd,
            double maxRideTime,
            VehicleRoadRestrictionProperties roadProperties,
            VehicleType type,
            double vehicleCostPerDistanceUnit,
            double vehicleCostPerTimeUnit,
            double vehicleCostPerUsage,
            int ownerID )
```

Creates vehicle object for backward compatibility

Parameters

id	
capacity	
specialProperties	
capacityAggregation	
initialLocation	
availabilityStart	
finalLocation	
availabilityEnd	
maxRideTime	
roadProperties	
type	
vehicleCostPerDistanceUnit	
vehicleCostPerTimeUnit	
vehicleCostPerUsage	
ownerID	

Definition at line 94 of file Vehicle.cs.

6.20.2.2 Vehicle() [2/2]

```
VRPTWOptimizer.Vehicle.Vehicle (
            int id,
            double[] capacity,
            int[] specialProperties,
            Aggregation[] capacityAggregation,
            Location initialLocation,
            double availabilityStart,
            Location finalLocation,
            double availabilityEnd,
            double maxRideTime,
            VehicleRoadRestrictionProperties roadProperties,
            VehicleType type,
            double vehicleCostPerDistanceUnit,
            double vehicleCostPerTimeUnit,
            double vehicleCostPerUsage,
            int ownerID,
            double vehicleFlatCostForShortRouteLength,
            double vehicleMaxRouteLengthForFlatCost )
```

Creates vehicle object

Parameters

id	
capacity	
specialProperties	
capacityAggregation	
initialLocation	

Parameters

availabilityStart	
finalLocation	
availabilityEnd	
maxRideTime	
roadProperties	
type	
vehicleCostPerDistanceUnit	
vehicleCostPerTimeUnit	
vehicleCostPerUsage	
ownerID	
vehicleFlatCostForShortRouteLength	
vehicleMaxRouteLengthForFlatCost	

Definition at line 148 of file Vehicle.cs.

6.20.3 Member Function Documentation

6.20.3.1 CanFitCapacity()

```
static bool VRPTWOptimizer.Vehicle.CanFitCapacity (
            double[] capacity,
            Aggregation[] aggregationType,
            IEnumerable< double[]> sizes )  [static]
```

Checks if the given list of cargo sizes would fit into specified capacity according to proper size aggregation

Parameters

capacity	
aggregationType	
sizes	

Returns

Definition at line 226 of file Vehicle.cs.

6.20.3.2 CanFitRequests()

```
bool VRPTWOptimizer.Vehicle.CanFitRequests (
            IEnumerable< TransportRequest > requests )
```

Verifies if a pool of TransportRequest objects can fit together within the Vehicle

Parameters

requests	

Returns

Definition at line 257 of file Vehicle.cs.

6.20.3.3 CanFitRequestsSomewhereInVehicle()

```
bool VRPTWOptimizer.Vehicle.CanFitRequestsSomewhereInVehicle (
            IEnumerable< TransportRequest > requests )
```

Verifies if a pool of TransportRequest objects can fit together within the Vehicle

Parameters

requests	

Returns

Definition at line 279 of file Vehicle.cs.

6.20.3.4 CanHandleRequest()

```
bool VRPTWOptimizer.Vehicle.CanHandleRequest (
            TransportRequest candidateRequest )
```

Verifies if properties of TransportRequest conform with abilities of Vehicle and restrictions of TransportRequest allow the Vehicle to approach initial and final destinations

Parameters

candidateRequest	

Returns

Definition at line 317 of file Vehicle.cs.

6.20.4 Property Documentation

6.20.4.1 Capacity

```
double [] VRPTWOptimizer.Vehicle.Capacity  [get], [protected set]
```

Array of capacity dimensions (e.g. mass, length, volume, euro pallets count) Freezer availability can also be treated as dimension

Definition at line 19 of file Vehicle.cs.

6.20.4.2 CapacityAggregationType

```
Aggregation [] VRPTWOptimizer.Vehicle.CapacityAggregationType  [get], [protected set]
```

Array of how the packages sizes are aggregated in various dimensions (e.g. sum for mass, max for length)

Definition at line 23 of file Vehicle.cs.

6.20.4.3 FinalLocation

```
Location VRPTWOptimizer.Vehicle.FinalLocation  [get], [protected set]
```

Location where vehicle needs to finish its operations

Definition at line 27 of file Vehicle.cs.

6.20.4.4 InitialLocation

```
Location VRPTWOptimizer.Vehicle.InitialLocation  [get], [protected set]
```

Location where vehicle will be available at AvailabilityStart

Definition at line 31 of file Vehicle.cs.

6.20.4.5 MaxRideTime

```
double VRPTWOptimizer.Vehicle.MaxRideTime  [get], [protected set]
```

Max time span when vehicle can be on road

Definition at line 35 of file Vehicle.cs.

6.20.4.6 OwnerID

```
int VRPTWOptimizer.Vehicle.OwnerID  [get], [protected set]
```

Identifier of the company owning the vehicle

Definition at line 39 of file Vehicle.cs.

6.20.4.7 RoadProperties

```
VehicleRoadRestrictionProperties VRPTWOptimizer.Vehicle.RoadProperties  [get], [protected set]
```

Size of the vehicle that affects possibility of traversing given route or accessing a certain Location

Definition at line 43 of file Vehicle.cs.

6.20.4.8 SpecialProperties

```
int [] VRPTWOptimizer.Vehicle.SpecialProperties  [get], [protected set]
```

Describes properties of vehicles within a given domain (e.g. lift, freezer)

Definition at line 47 of file Vehicle.cs.

6.20.4.9 Type

```
VehicleType VRPTWOptimizer.Vehicle.Type  [get], [protected set]
```

Type of vehicle necessary to decide if needs to be combined with a unit with engine (semi-trailer) or not (truck)

Definition at line 51 of file Vehicle.cs.

6.20.4.10 VehicleCostPerDistanceUnit

```
double VRPTWOptimizer.Vehicle.VehicleCostPerDistanceUnit [get], [protected set]
```

Cost of using vehicle per distance unit

Definition at line 55 of file Vehicle.cs.

6.20.4.11 VehicleCostPerTimeUnit

```
double VRPTWOptimizer.Vehicle.VehicleCostPerTimeUnit [get], [protected set]
```

Cost of using vehicle per time unit (while moving from Location to Location)

Definition at line 59 of file Vehicle.cs.

6.20.4.12 VehicleCostPerUsage

```
double VRPTWOptimizer.Vehicle.VehicleCostPerUsage [get], [protected set]
```

Cost of using vehicle at all in a given problem

Definition at line 63 of file Vehicle.cs.

6.20.4.13 VehicleFlatCostForShortRouteLength

```
double VRPTWOptimizer.Vehicle.VehicleFlatCostForShortRouteLength [get]
```

Cost of using vehicle if route length is shorter than VehicleMaxRouteLengthForFlatCost (in meters)

Definition at line 69 of file Vehicle.cs.

6.20.4.14 VehicleMaxRouteLengthForFlatCost

```
double VRPTWOptimizer.Vehicle.VehicleMaxRouteLengthForFlatCost [get]
```

Distance (by default in meters) for which VehicleFlatCostForShortRouteLength flat rate is applied, otherwise VehicleCostPerDistanceUnit $*$ Distance is applieds

Definition at line 74 of file Vehicle.cs.

The documentation for this class was generated from the following file:

- Vehicle.cs

6.21 VRPTWOptimizer.Dto.VehicleSchedule Class Reference

Schedules vehicle presence at warehouse grounds

Properties

- VehicleType CapacityVehicleType `[get, set]`

  Type of Vehicle (truck or semi-trailer)
- int EpCapacity `[get, set]`

  Size of the vehicle in europallets
- int VehicleId `[get, set]`

  Id of the vehicle
- List< TimeInterval > YardAvailabilitySchedule `[get, set]`

  List of intervals when the vehicle is planned to be available for the warehouse to operate on (wash, refuel, load or park on yard)

6.21.1 Detailed Description

Schedules vehicle presence at warehouse grounds

Definition at line 14 of file VehicleSchedule.cs.

6.21.2 Property Documentation

6.21.2.1 CapacityVehicleType

```
VehicleType VRPTWOptimizer.Dto.VehicleSchedule.CapacityVehicleType  [get], [set]
```

Type of Vehicle (truck or semi-trailer)

Definition at line 20 of file VehicleSchedule.cs.

6.21.2.2 EpCapacity

```
int VRPTWOptimizer.Dto.VehicleSchedule.EpCapacity  [get], [set]
```

Size of the vehicle in europallets

Definition at line 25 of file VehicleSchedule.cs.

6.21.2.3 VehicleId

```
int VRPTWOptimizer.Dto.VehicleSchedule.VehicleId  [get], [set]
```

Id of the vehicle

Definition at line 30 of file VehicleSchedule.cs.

6.21.2.4 YardAvailabilitySchedule

```
List<TimeInterval> VRPTWOptimizer.Dto.VehicleSchedule.YardAvailabilitySchedule  [get], [set]
```

List of intervals when the vehicle is planned to be available for the warehouse to operate on (wash, refuel, load or park on yard)

Definition at line 35 of file VehicleSchedule.cs.

The documentation for this class was generated from the following file:

- VehicleSchedule.cs

6.22 VRPTWOptimizer.VRPCostFunction Class Reference

Class for calculating solution costs

Public Member Functions

- VRPCostFunction ()

  Constructor for deserializer
- VRPCostFunction (double distanceFactor, double usageFactor, double driveTimeFactor, double leftUnit↩ Factor, double maxDelaySquaredFactor, double maxEarlyArrivalFactor, double totalDelaySquaredFactor, double totalEarlyArrivalFactor, double carrierMinDistanceFactor, double carrierShareFactor, double fillIn↩ Factor, Dictionary< int, double > carrierMinDistanceThreshold, Dictionary< int, double > carrierShareRatio, double routesCountFactor=0, double totalEarlyArrivalSquaredFactor=0, double totalDelayFactor=0, double maxVehicleSpreadFactor=0, double maxEarlyArrivalSquaredFactor=0, double maxDelayFactor=0)

  Creates cost function object with the specified weight for given aspects of problem solution
- double SingleRouteValue (IRoute route)

  Evaluate cost of a single route
- double Value (List< IRoute > routes, List< TransportRequest > leftRequests)

  Evaluate cost of all routes

Static Public Member Functions

- static double ComputeFillInFactor (List< double[ ]> cargoOnRouteStart, double[ ] vehicleCapacity, Enums.Aggregation[ ] aggregationType)

  Computes fill in factor at route start
- static double ComputeFillInFactor (IRoute route)

  Computes fill in factor for given IRoute
- static double ComputeMaxEarlyArrival (IRoute route)

  Computes earliest arrival before preferred time window start
- static double ComputeMaxTimeDiff (double[ ] referenceValues, double[ ] trueValues, bool referenceIsLowerBound)

  Computes max unwanted time difference
- static double ComputeTotalEarlyArrival (IRoute route)

  Computes sum of early arrivals within the route
- static double ComputeTotalTimeDiff (double[ ] referenceValues, double[ ] trueValues, bool referenceIsLowerBound)

  Computes total unwanted time difference
- static VRPCostFunction GetDefaultParametersFunction ()

  Creates cost function with parameters prioritizing in following order

Properties

- double CarrierMinDistanceFactor `[get, set]`

  Weight of not getting min distance per carrier
- Dictionary< int, double > CarrierMinDistanceThreshold `[get, set]`

  Min distance per carrier
- double CarrierShareFactor `[get, set]`

  Weight of not getting desired division of distance between carriers
- Dictionary< int, double > CarrierShareRatio `[get, set]`

  Desired distances division between carriers
- double DistanceFactor `[get, set]`

  Weight of the distance costs in final route evaluations
- double DriveTimeFactor `[get, set]`

  Weight of the drive time costs in final route evaluations
- double FillInFactor `[get, set]`

  Weight of sending empty vehicles out of the depot
- double LeftCargoUnitFactor `[get, set]`

  Weight for not delivering a single unit of cargo
- double MaxDelayFactor `[get, set]`

  Weight for max delay time en route
- double MaxDelaySquaredFactor `[get, set]`

  Weight for squared max delay time en route
- double MaxEarlyArrivalFactor `[get, set]`

  Weight for max early arrival en route
- double MaxEarlyArrivalSquaredFactor `[get, set]`

  Weight for squared max early arrival en route
- double MaxVehicleSpreadFactor `[get, set]`

  Weight for max vehicle wait time in depot
- double RoutesCountFactor `[get, set]`

  Weight of the number of routes in final evaluation
- double TotalDelayFactor `[get, set]`

Weight for total delay time en route

- double TotalDelaySquaredFactor `[get, set]`

  Weight for squared total delay time en route
- double TotalEarlyArrivalFactor `[get, set]`

  Weight for total early arrival en route
- double TotalEarlyArrivalSquaredFactor `[get, set]`

  Weight for squared total early arrival en route
- double UsageFactor `[get, set]`

  Weight of the vehicle usage costs in final route evaluations

6.22.1 Detailed Description

Class for calculating solution costs

Definition at line 12 of file VRPCostFunction.cs.

6.22.2 Constructor & Destructor Documentation

6.22.2.1 VRPCostFunction() [1/2]

```
VRPTWOptimizer.VRPCostFunction.VRPCostFunction ( )
```

Constructor for deserializer

Definition at line 99 of file VRPCostFunction.cs.

6.22.2.2 VRPCostFunction() [2/2]

```
VRPTWOptimizer.VRPCostFunction.VRPCostFunction (
            double distanceFactor,
            double usageFactor,
            double driveTimeFactor,
            double leftUnitFactor,
            double maxDelaySquaredFactor,
            double maxEarlyArrivalFactor,
            double totalDelaySquaredFactor,
            double totalEarlyArrivalFactor,
            double carrierMinDistanceFactor,
            double carrierShareFactor,
            double fillInFactor,
            Dictionary< int, double > carrierMinDistanceThreshold,
            Dictionary< int, double > carrierShareRatio,
            double routesCountFactor = 0,
            double totalEarlyArrivalSquaredFactor = 0,
            double totalDelayFactor = 0,
            double maxVehicleSpreadFactor = 0,
            double maxEarlyArrivalSquaredFactor = 0,
            double maxDelayFactor = 0 )
```

Creates cost function object with the specified weight for given aspects of problem solution

Parameters

distanceFactor	
usageFactor	
driveTimeFactor	
leftUnitFactor	
maxDelaySquaredFactor	
maxEarlyArrivalFactor	
totalDelaySquaredFactor	
totalEarlyArrivalFactor	
carrierMinDistanceFactor	
carrierShareFactor	
fillInFactor	
carrierMinDistanceThreshold	
carrierShareRatio	
routesCountFactor	
totalEarlyArrivalSquaredFactor	
totalDelayFactor	
maxVehicleSpreadFactor	
maxEarlyArrivalSquaredFactor	
maxDelayFactor	

Definition at line 133 of file VRPCostFunction.cs.

6.22.3 Member Function Documentation

6.22.3.1 ComputeFillInFactor() [1/2]

```
static double VRPTWOptimizer.VRPCostFunction.ComputeFillInFactor (
            IRoute route )  [static]
```

Computes fill in factor for given IRoute

Parameters

route	

Returns

Definition at line 208 of file VRPCostFunction.cs.

6.22.3.2 ComputeFillInFactor() [2/2]

```
static double VRPTWOptimizer.VRPCostFunction.ComputeFillInFactor (
            List< double[]> cargoOnRouteStart,
            double[] vehicleCapacity,
            Enums.Aggregation[] aggregationType )  [static]
```

Computes fill in factor at route start

Parameters

cargoOnRouteStart	
vehicleCapacity	

Returns

Definition at line 181 of file VRPCostFunction.cs.

6.22.3.3 ComputeMaxEarlyArrival()

```
static double VRPTWOptimizer.VRPCostFunction.ComputeMaxEarlyArrival (
            IRoute route )  [static]
```

Computes earliest arrival before preferred time window start

Parameters

route	

Returns

Definition at line 221 of file VRPCostFunction.cs.

6.22.3.4 ComputeMaxTimeDiff()

```
static double VRPTWOptimizer.VRPCostFunction.ComputeMaxTimeDiff (
            double[] referenceValues,
            double[] trueValues,
            bool referenceIsLowerBound )  [static]
```

Computes max unwanted time difference

Parameters

referenceValues	Bound values
trueValues	Real values
referenceIsLowerBound	Is reference a lower bound for true value?

Returns

Definition at line 237 of file VRPCostFunction.cs.

6.22.3.5 ComputeTotalEarlyArrival()

```
static double VRPTWOptimizer.VRPCostFunction.ComputeTotalEarlyArrival (
            IRoute route )  [static]
```

Computes sum of early arrivals within the route

Parameters

route	

Returns

Definition at line 259 of file VRPCostFunction.cs.

6.22.3.6 ComputeTotalTimeDiff()

```
static double VRPTWOptimizer.VRPCostFunction.ComputeTotalTimeDiff (
            double[] referenceValues,
            double[] trueValues,
            bool referenceIsLowerBound )  [static]
```

Computes total unwanted time difference

Parameters

referenceValues	Bound values
trueValues	Real values
referenceIsLowerBound	Is reference a lower bound for true value?

Returns

Definition at line 275 of file VRPCostFunction.cs.

6.22.3.7 GetDefaultParametersFunction()

```
static VRPCostFunction VRPTWOptimizer.VRPCostFunction.GetDefaultParametersFunction ( )  [static]
```

Creates cost function with parameters prioritizing in following order

- not leaving any cargo
- small number of routes
- balanced sum of distance, delays and early arrivals

Returns

Definition at line 299 of file VRPCostFunction.cs.

6.22.3.8 SingleRouteValue()

```
double VRPTWOptimizer.VRPCostFunction.SingleRouteValue (
            IRoute route )
```

Evaluate cost of a single route

Parameters

route	

Returns

Definition at line 324 of file VRPCostFunction.cs.

6.22.3.9 Value()

```
double VRPTWOptimizer.VRPCostFunction.Value (
            List< IRoute > routes,
            List< TransportRequest > leftRequests )
```

Evaluate cost of all routes

Parameters

routes	
leftRequests	

Returns

Definition at line 366 of file VRPCostFunction.cs.

6.22.4 Property Documentation

6.22.4.1 CarrierMinDistanceFactor

```
double VRPTWOptimizer.VRPCostFunction.CarrierMinDistanceFactor  [get], [set]
```

Weight of not getting min distance per carrier

Definition at line 22 of file VRPCostFunction.cs.

6.22.4.2 CarrierMinDistanceThreshold

```
Dictionary<int, double> VRPTWOptimizer.VRPCostFunction.CarrierMinDistanceThreshold  [get],
[set]
```

Min distance per carrier

Definition at line 26 of file VRPCostFunction.cs.

6.22.4.3 CarrierShareFactor

```
double VRPTWOptimizer.VRPCostFunction.CarrierShareFactor  [get], [set]
```

Weight of not getting desired division of distance between carriers

Definition at line 30 of file VRPCostFunction.cs.

6.22.4.4 CarrierShareRatio

```
Dictionary<int, double> VRPTWOptimizer.VRPCostFunction.CarrierShareRatio  [get], [set]
```

Desired distances division between carriers

Definition at line 34 of file VRPCostFunction.cs.

6.22.4.5 DistanceFactor

```
double VRPTWOptimizer.VRPCostFunction.DistanceFactor  [get], [set]
```

Weight of the distance costs in final route evaluations

Definition at line 38 of file VRPCostFunction.cs.

6.22.4.6 DriveTimeFactor

```
double VRPTWOptimizer.VRPCostFunction.DriveTimeFactor  [get], [set]
```

Weight of the drive time costs in final route evaluations

Definition at line 42 of file VRPCostFunction.cs.

6.22.4.7 FillInFactor

```
double VRPTWOptimizer.VRPCostFunction.FillInFactor  [get], [set]
```

Weight of sending empty vehicles out of the depot

Definition at line 46 of file VRPCostFunction.cs.

6.22.4.8 LeftCargoUnitFactor

```
double VRPTWOptimizer.VRPCostFunction.LeftCargoUnitFactor  [get], [set]
```

Weight for not delivering a single unit of cargo

Definition at line 50 of file VRPCostFunction.cs.

6.22.4.9 MaxDelayFactor

```
double VRPTWOptimizer.VRPCostFunction.MaxDelayFactor  [get], [set]
```

Weight for max delay time en route

Definition at line 54 of file VRPCostFunction.cs.

6.22.4.10 MaxDelaySquaredFactor

```
double VRPTWOptimizer.VRPCostFunction.MaxDelaySquaredFactor  [get], [set]
```

Weight for squared max delay time en route

Definition at line 58 of file VRPCostFunction.cs.

6.22.4.11 MaxEarlyArrivalFactor

```
double VRPTWOptimizer.VRPCostFunction.MaxEarlyArrivalFactor  [get], [set]
```

Weight for max early arrival en route

Definition at line 62 of file VRPCostFunction.cs.

6.22.4.12 MaxEarlyArrivalSquaredFactor

```
double VRPTWOptimizer.VRPCostFunction.MaxEarlyArrivalSquaredFactor  [get], [set]
```

Weight for squared max early arrival en route

Definition at line 66 of file VRPCostFunction.cs.

6.22.4.13 MaxVehicleSpreadFactor

```
double VRPTWOptimizer.VRPCostFunction.MaxVehicleSpreadFactor  [get], [set]
```

Weight for max vehicle wait time in depot

Definition at line 70 of file VRPCostFunction.cs.

6.22.4.14 RoutesCountFactor

```
double VRPTWOptimizer.VRPCostFunction.RoutesCountFactor  [get], [set]
```

Weight of the number of routes in final evaluation

Definition at line 74 of file VRPCostFunction.cs.

6.22.4.15 TotalDelayFactor

```
double VRPTWOptimizer.VRPCostFunction.TotalDelayFactor  [get], [set]
```

Weight for total delay time en route

Definition at line 78 of file VRPCostFunction.cs.

6.22.4.16 TotalDelaySquaredFactor

```
double VRPTWOptimizer.VRPCostFunction.TotalDelaySquaredFactor  [get], [set]
```

Weight for squared total delay time en route

Definition at line 82 of file VRPCostFunction.cs.

6.22.4.17 TotalEarlyArrivalFactor

```
double VRPTWOptimizer.VRPCostFunction.TotalEarlyArrivalFactor  [get], [set]
```

Weight for total early arrival en route

Definition at line 86 of file VRPCostFunction.cs.

6.22.4.18 TotalEarlyArrivalSquaredFactor

```
double VRPTWOptimizer.VRPCostFunction.TotalEarlyArrivalSquaredFactor  [get], [set]
```

Weight for squared total early arrival en route

Definition at line 90 of file VRPCostFunction.cs.

6.22.4.19 UsageFactor

```
double VRPTWOptimizer.VRPCostFunction.UsageFactor [get], [set]
```

Weight of the vehicle usage costs in final route evaluations

Definition at line 94 of file VRPCostFunction.cs.

The documentation for this class was generated from the following file:

- VRPCostFunction.cs

6.23 VRPTWOptimizer.VRPDefinition Class Reference

Describes data for a generalized Vehicle Routing Problem

Public Member Functions

- void AddSolution (VRPSolution vrpSolution)

  Adds solution to the collection of problem solutions
- string ToPrettyJSONString ()

  Generates indended JSON definition of VRP
- bool TrySaveToFile (string filename)

  Writes VRPDefinition and VRPSolutions list to a JSON file using pretty formatter

Static Public Member Functions

- static VRPDefinition GenerateVRPDefintion (IVRPProvider vrpProvider, DateTime billingDate, IDistance↩ Provider distanceProvider, ITimeEstimator timeEstimator, string client)

  Creates VRPProblemDefinition standard format from problem description
- static VRPDefinition GenerateVRPDefintion (IVRPProvider vrpProvider, VRPCostFunction costFunction↩ Factors, DateTime billingDate, IDistanceProvider distanceProvider, ITimeEstimator timeEstimator, string client)

  Creates VRPProblemDefinition standard format from problem description

Properties

- string Client `[get, set]`

  Description of the company seeking solution to VRP problem
- VRPCostFunction CostFunctionFactors `[get, set]`

  Factors to be taken into account while optimizing the problem
- Version DataFormatVersion `[get]`

  Version of the library containg data format definition
- DateTime Date `[get, set]`

  Date for which the problem is solved
- string DepotId `[get, set]`

  Depot for which the problem is solved
- IDistanceProvider DistanceData `[get, set]`

Data for computing distances between points (straightforward matrix, formula description, API access information)

- List< Driver > Drivers `[get, set]`

  List of drivers
- List< TransportRequest > Requests `[get, set]`

  List of TransportRequests from given day in given depot
- ITimeEstimator ServiceTimeEstimator `[get, set]`

  Object describing parameters of service (loading/unloading) time estimator model
- List< VRPSolution > Solutions `[get, set]`

  List of solutions (vehicles assignments and schedule)
- List< Vehicle > Vehicles `[get, set]`

  List of available vehicles
- DateTime ZeroHour `[get, set]`

  Real time value to computer relative seconds against to retrieve real timestamps

6.23.1 Detailed Description

Describes data for a generalized Vehicle Routing Problem

Definition at line 14 of file VRPDefinition.cs.

6.23.2 Member Function Documentation

6.23.2.1 AddSolution()

```
void VRPTWOptimizer.VRPDefinition.AddSolution (
            VRPSolution vrpSolution )
```

Adds solution to the collection of problem solutions

Parameters

vrpSolution	

Definition at line 173 of file VRPDefinition.cs.

6.23.2.2 GenerateVRPDefintion() [1/2]

```
static VRPDefinition VRPTWOptimizer.VRPDefinition.GenerateVRPDefintion (
            IVRPProvider vrpProvider,
            DateTime billingDate,
            IDistanceProvider distanceProvider,
            ITimeEstimator timeEstimator,
            string client )  [static]
```

Creates VRPProblemDefinition standard format from problem description

Parameters

vrpProvider	
billingDate	
distanceProvider	
timeEstimator	
client	

Returns

Definition at line 115 of file VRPDefinition.cs.

6.23.2.3 GenerateVRPDefintion() [2/2]

```
static VRPDefinition VRPTWOptimizer.VRPDefinition.GenerateVRPDefintion (
            IVRPProvider vrpProvider,
            VRPCostFunction costFunctionFactors,
            DateTime billingDate,
            IDistanceProvider distanceProvider,
            ITimeEstimator timeEstimator,
            string client )  [static]
```

Creates VRPProblemDefinition standard format from problem description

Parameters

vrpProvider	
costFunctionFactors	
billingDate	
distanceProvider	
timeEstimator	
client	

Returns

Definition at line 142 of file VRPDefinition.cs.

6.23.2.4 ToPrettyJSONString()

```
string VRPTWOptimizer.VRPDefinition.ToPrettyJSONString ( )
```

Generates indended JSON definition of VRP

Returns

Definition at line 186 of file VRPDefinition.cs.

6.23.2.5 TrySaveToFile()

```
bool VRPTWOptimizer.VRPDefinition.TrySaveToFile (
            string filename )
```

Writes VRPDefinition and VRPSolutions list to a JSON file using pretty formatter

Parameters

filename	

Returns

Definition at line 203 of file VRPDefinition.cs.

6.23.3 Property Documentation

6.23.3.1 Client

```
string VRPTWOptimizer.VRPDefinition.Client  [get], [set]
```

Description of the company seeking solution to VRP problem

Definition at line 58 of file VRPDefinition.cs.

6.23.3.2 CostFunctionFactors

```
VRPCostFunction VRPTWOptimizer.VRPDefinition.CostFunctionFactors  [get], [set]
```

Factors to be taken into account while optimizing the problem

Definition at line 62 of file VRPDefinition.cs.

6.23.3.3 DataFormatVersion

```
Version VRPTWOptimizer.VRPDefinition.DataFormatVersion  [get]
```

Version of the library containg data format definition

Definition at line 66 of file VRPDefinition.cs.

6.23.3.4 Date

```
DateTime VRPTWOptimizer.VRPDefinition.Date  [get], [set]
```

Date for which the problem is solved

Definition at line 71 of file VRPDefinition.cs.

6.23.3.5 DepotId

```
string VRPTWOptimizer.VRPDefinition.DepotId  [get], [set]
```

Depot for which the problem is solved

Definition at line 75 of file VRPDefinition.cs.

6.23.3.6 DistanceData

```
IDistanceProvider VRPTWOptimizer.VRPDefinition.DistanceData  [get], [set]
```

Data for computing distances between points (straightforward matrix, formula description, API access information)

Definition at line 79 of file VRPDefinition.cs.

6.23.3.7 Drivers

```
List<Driver> VRPTWOptimizer.VRPDefinition.Drivers  [get], [set]
```

List of drivers

Definition at line 83 of file VRPDefinition.cs.

6.23.3.8 Requests

```
List<TransportRequest> VRPTWOptimizer.VRPDefinition.Requests  [get], [set]
```

List of TransportRequests from given day in given depot

Definition at line 87 of file VRPDefinition.cs.

6.23.3.9 ServiceTimeEstimator

```
ITimeEstimator VRPTWOptimizer.VRPDefinition.ServiceTimeEstimator  [get], [set]
```

Object describing parameters of service (loading/unloading) time estimator model

Definition at line 91 of file VRPDefinition.cs.

6.23.3.10 Solutions

```
List<VRPSolution> VRPTWOptimizer.VRPDefinition.Solutions  [get], [set]
```

List of solutions (vehicles assignments and schedule)

Definition at line 95 of file VRPDefinition.cs.

6.23.3.11 Vehicles

```
List<Vehicle> VRPTWOptimizer.VRPDefinition.Vehicles  [get], [set]
```

List of available vehicles

Definition at line 99 of file VRPDefinition.cs.

6.23.3.12 ZeroHour

```
DateTime VRPTWOptimizer.VRPDefinition.ZeroHour  [get], [set]
```

Real time value to computer relative seconds against to retrieve real timestamps

Definition at line 103 of file VRPDefinition.cs.

The documentation for this class was generated from the following file:

- VRPDefinition.cs

6.24 VRPTWOptimizer.VRPOptimizerResult Class Reference

Represents the output of Vehicle Routing Problem optimization algorithm

Properties

- List< TransportRequest > LeftRequests [get, set]

  List of TransportRequest that the algorithm was unable to fit into any of routes
- List< IRoute > Routes [get, set]

  Visits schedule for the Vehicle objects

6.24.1 Detailed Description

Represents the output of Vehicle Routing Problem optimization algorithm

Definition at line 9 of file VRPOptimizerResult.cs.

6.24.2 Property Documentation

6.24.2.1 LeftRequests

```
List<TransportRequest> VRPTWOptimizer.VRPOptimizerResult.LeftRequests [get], [set]
```

List of TransportRequest that the algorithm was unable to fit into any of routes

Definition at line 14 of file VRPOptimizerResult.cs.

6.24.2.2 Routes

```
List<IRoute> VRPTWOptimizer.VRPOptimizerResult.Routes [get], [set]
```

Visits schedule for the Vehicle objects

Definition at line 19 of file VRPOptimizerResult.cs.

The documentation for this class was generated from the following file:

- VRPOptimizerResult.cs

6.25 VRPTWOptimizer.Interfaces.VRPResult< R, V, Rt > Class Template Reference

Class containing list of routes for each tractor and straight truck

Public Member Functions

- VRPResult (Dictionary< V, List< Rt > > routes, List< R > leftRequests)

  Creates VRP results from routes dictionary and left requests list

Static Public Member Functions

- static implicit operator Tuple< List< Rt >, List< R > > (VRPResult< R, V, Rt > vrp)

  Converts solution into Tuple for backward compatibility

Properties

- List< R > LeftRequests `[get]`

  Requests that were not assigned to any vehicle
- List< Rt > Routes `[get]`

  Flat list of routes without tractor assignment
- Dictionary< V, List< Rt > > TractorRoutes `[get]`

  Full schedule with routes for each tractor and straight truck

6.25.1 Detailed Description

Class containing list of routes for each tractor and straight truck

Template Parameters

R	
V	
Rt	

Type Constraints

R **:** ***TransportRequest***

V **:** ***Vehicle***

Rt **:** ***IRoute***

Definition at line 13 of file VRPResult.cs.

6.25.2 Constructor & Destructor Documentation

6.25.2.1 VRPResult()

```
VRPTWOptimizer.Interfaces.VRPResult< R, V, Rt >.VRPResult (
            Dictionary< V, List< Rt > > routes,
            List< R > leftRequests )
```

Creates VRP results from routes dictionary and left requests list

Parameters

routes	
leftRequests	

Definition at line 38 of file VRPResult.cs.

6.25.3 Member Function Documentation

6.25.3.1 operator Tuple< List< Rt >, List< R > >()

```
static implicit VRPTWOptimizer.Interfaces.VRPResult< R, V, Rt >.operator Tuple< List< Rt >,
List< R > > (
            VRPResult< R, V, Rt > vrp )  [static]
```

Converts solution into Tuple for backward compatibility

Parameters

vrp	

6.25.4 Property Documentation

6.25.4.1 LeftRequests

```
List<R> VRPTWOptimizer.Interfaces.VRPResult< R, V, Rt >.LeftRequests  [get]
```

Requests that were not assigned to any vehicle

Definition at line 21 of file VRPResult.cs.

6.25.4.2 Routes

```
List<Rt> VRPTWOptimizer.Interfaces.VRPResult< R, V, Rt >.Routes [get]
```

Flat list of routes without tractor assignment

Definition at line 27 of file VRPResult.cs.

6.25.4.3 TractorRoutes

```
Dictionary<V, List<Rt> > VRPTWOptimizer.Interfaces.VRPResult< R, V, Rt >.TractorRoutes [get]
```

Full schedule with routes for each tractor and straight truck

Definition at line 31 of file VRPResult.cs.

The documentation for this class was generated from the following file:

- VRPResult.cs

6.26 VRPTWOptimizer.VRPSolution Class Reference

Definition of structure describing solution to the Vehicle Routing Problem. Includes Vehicle assignment to TransportRequest and Vehicle schedule

Classes

- class ScheduleItem

  Single time entry describing planned visit at a given Location
- class TransportItem

  Entry describing a single loop of the Vehicle/combined Vehicle

Static Public Member Functions

- static VRPSolution GenerateVRPSolution (IVRPOptimizer _optimizer, DateTime computationsStart, DateTime computationsEnd, List< TransportRequest > leftRequests, List< IRoute > routes)

  Creates VRPSolution in a standard format on the basis of results and computation properties

Properties

- string Algorithm [get, set]

  Name of the algorithm that generated this solution
- double ComputationTime [get, set]

  Computations time it took to generate the solution (problem data is assumed to be loaded to memory)
- DateTime ComputationTimestamp [get, set]

  Timestamp when the computations were being performed
- string ComputerId [get, set]

  NETBIOS computer name that run the computations
- int DelaysCount [get, set]

  Number of transport requests that were late (even 1 second against the prefered time)
- List< int > LeftRequestsIds [get, set]

  Ids of the TransportRequest objects that were not assigned to any Vehicle
- double MaxDelay [get, set]

  Max delay in serving an assigned TransportRequest (against the prefered time)
- double TotalDelay [get, set]

  Sum of all delay in serving TransportRequest objects (against the prefered time)
- double TotalLength [get, set]

  Length of all routes in meters
- List< TransportItem > Transports [get, set]

  List of all assigned transports
- Version Version [get, set]

  Algorithm library version

6.26.1 Detailed Description

Definition of structure describing solution to the Vehicle Routing Problem. Includes Vehicle assignment to TransportRequest and Vehicle schedule

Definition at line 13 of file VRPSolution.cs.

6.26.2 Member Function Documentation

6.26.2.1 GenerateVRPSolution()

```
static VRPSolution VRPTWOptimizer.VRPSolution.GenerateVRPSolution (
            IVRPOptimizer _optimizer,
            DateTime computationsStart,
            DateTime computationsEnd,
            List< TransportRequest > leftRequests,
            List< IRoute > routes )  [static]
```

Creates VRPSolution in a standard format on the basis of results and computation properties

Parameters

_optimizer	
computationsStart	
computationsEnd	
leftRequests	
routes	

Returns

Definition at line 145 of file VRPSolution.cs.

6.26.3 Property Documentation

6.26.3.1 Algorithm

```
string VRPTWOptimizer.VRPSolution.Algorithm  [get], [set]
```

Name of the algorithm that generated this solution

Definition at line 94 of file VRPSolution.cs.

6.26.3.2 ComputationTime

```
double VRPTWOptimizer.VRPSolution.ComputationTime  [get], [set]
```

Computations time it took to generate the solution (problem data is assumed to be loaded to memory)

Definition at line 98 of file VRPSolution.cs.

6.26.3.3 ComputationTimestamp

```
DateTime VRPTWOptimizer.VRPSolution.ComputationTimestamp  [get], [set]
```

Timestamp when the computations were being performed

Definition at line 102 of file VRPSolution.cs.

6.26.3.4 ComputerId

```
string VRPTWOptimizer.VRPSolution.ComputerId  [get], [set]
```

NETBIOS computer name that run the computations

Definition at line 106 of file VRPSolution.cs.

6.26.3.5 DelaysCount

```
int VRPTWOptimizer.VRPSolution.DelaysCount  [get], [set]
```

Number of transport requests that were late (even 1 second against the prefered time)

Definition at line 110 of file VRPSolution.cs.

6.26.3.6 LeftRequestsIds

```
List<int> VRPTWOptimizer.VRPSolution.LeftRequestsIds  [get], [set]
```

Ids of the TransportRequest objects that were not assigned to any Vehicle

Definition at line 114 of file VRPSolution.cs.

6.26.3.7 MaxDelay

```
double VRPTWOptimizer.VRPSolution.MaxDelay  [get], [set]
```

Max delay in serving an assigned TransportRequest (against the prefered time)

Definition at line 118 of file VRPSolution.cs.

6.26.3.8 TotalDelay

```
double VRPTWOptimizer.VRPSolution.TotalDelay  [get], [set]
```

Sum of all delay in serving TransportRequest objects (against the prefered time)

Definition at line 122 of file VRPSolution.cs.

6.26.3.9 TotalLength

```
double VRPTWOptimizer.VRPSolution.TotalLength  [get], [set]
```

Length of all routes in meters

Definition at line 126 of file VRPSolution.cs.

6.26.3.10 Transports

```
List<TransportItem> VRPTWOptimizer.VRPSolution.Transports  [get], [set]
```

List of all assigned transports

Definition at line 130 of file VRPSolution.cs.

6.26.3.11 Version

```
Version VRPTWOptimizer.VRPSolution.Version  [get], [set]
```

Algorithm library version

Definition at line 134 of file VRPSolution.cs.

The documentation for this class was generated from the following file:

- VRPSolution.cs

Chapter 7

File Documentation

7.1 CargoUnit.cs File Reference

Classes

- class VRPTWOptimizer.CargoUnit

  Describes a single position on an order list (depending on the context it could be a europallet or single type of product with its quantity)

Namespaces

- namespace VRPTWOptimizer

7.2 CargoUnit.cs

Go to the documentation of this file.

```
00001 using Newtonsoft.Json;
00002 using System;
00003 using System.Collections.Generic;
00004 using System.Linq;
00005 using System.Text;
00006 using System.Threading.Tasks;
00007 using VRPTWOptimizer.Enums;
00008
00009 namespace VRPTWOptimizer
00010 {
00014     public class CargoUnit
00015     {
00019         [JsonProperty("zoneId")]
00020         public string CargoGroup { get; set; }
00024         [JsonProperty("cargoUnitType")]
00025         public CargoUnitType CargoUnitType { get; set; }
00029         [JsonProperty("goodId")]
00030         public int GoodsId { get; set; }
00034         [JsonProperty("goodName")]
00035         public string GoodsName { get; set; }
00039         [JsonProperty("importance")]
00040         public int Priority { get; set; }
00044         [JsonProperty("volume")]
00045         public double[] Size { get; set; }
00049         [JsonProperty("quantity")]
00050         public int UnitsCount { get; set; }
00051     }
00052 }
```

7.3 DictionaryDistanceProviderBase.cs File Reference

Classes

- class VRPTWOptimizer.DistanceProviders.DictionaryDistanceProviderBase

Namespaces

- namespace VRPTWOptimizer
- namespace VRPTWOptimizer.DistanceProviders

7.4 DictionaryDistanceProviderBase.cs

Go to the documentation of this file.

```
00001 using CommonGIS;
00002 using CommonGIS.Interfaces;
00003 using System.Collections.Generic;
00004 using System.Linq;
00005
00006 namespace VRPTWOptimizer.DistanceProviders
00007 {
00008     public abstract class DictionaryDistanceProviderBase :  IDistanceProvider
00009     {
00010         protected Dictionary<string, Dictionary<string, Dictionary<VehicleRoadRestrictionProperties,
      Distance»> distanceMatrix;
00011         protected Dictionary<VehicleRoadRestrictionProperties, VehicleRoadRestrictionProperties>
      vehicleToProfileMapper;
00012         protected bool SelfContain;
00013
00014         protected DictionaryDistanceProviderBase(bool selfContain)
00015         {
00016             StoredDistances = new List<Distance>();
00017             SelfContain = selfContain;
00018             distanceMatrix = new Dictionary<string, Dictionary<string,
      Dictionary<VehicleRoadRestrictionProperties, Distance»>();
00019             vehicleToProfileMapper = new Dictionary<VehicleRoadRestrictionProperties,
      VehicleRoadRestrictionProperties>(VehicleRoadRestrictionsComparer.Instance);
00020         }
00021
00022         public List<Distance> StoredDistances { get; set; }
00023
00024         protected void InitializeDistanceDictionary(List<Distance> distances)
00025         {
00026             StoredDistances.AddRange(distances);
00027
00028             foreach (var distance in StoredDistances)
00029             {
00030                 if (!distanceMatrix.ContainsKey(distance.FromId))
00031                 {
00032                     distanceMatrix.Add(distance.FromId, new Dictionary<string,
      Dictionary<VehicleRoadRestrictionProperties, Distance»());
00033                 }
00034                 if (!distanceMatrix[distance.FromId].ContainsKey(distance.ToId))
00035                 {
00036                     distanceMatrix[distance.FromId].Add(distance.ToId, new
      Dictionary<VehicleRoadRestrictionProperties, Distance>(VehicleRoadRestrictionsComparer.Instance));
00037                 }
00038                 if (!distanceMatrix[distance.FromId][distance.ToId].ContainsKey(distance.Profile))
00039                 {
00040                     distanceMatrix[distance.FromId][distance.ToId].Add(distance.Profile, distance);
00041                 }
00042             }
00043         }
00044
00045         public Distance GetDistance(
00046                     Location from,
00047             Location to,
00048             VehicleRoadRestrictionProperties vehicleProperties)
00049         {
00050             if (from.Id == to.Id)
00051             {
00052                 return new TimeLengthDistance(from.Id, to.Id, 0.0, 0.0, vehicleProperties);
00053             }
```

```
00054             else if (distanceMatrix.ContainsKey(from.Id) &&
      distanceMatrix[from.Id].ContainsKey(to.Id))
00055             {
00056                 if (!vehicleToProfileMapper.ContainsKey(vehicleProperties))
00057                 {
00058                     var profileCapacity = distanceMatrix[from.Id][to.Id].Keys
00059                         .Where(prf => prf.DoesVehicleFitIntoRestrictions(vehicleProperties))
00060                         .OrderBy(prf => prf.EpCount)
00061                         .OrderBy(prf => prf.GrossVehicleWeight)
00062                         .First();
00063                     vehicleToProfileMapper.Add(vehicleProperties, profileCapacity);
00064                 }
00065                 //TODO: zweryfikować którego dystansu właściwie brakuje
00066                 if
      (distanceMatrix[from.Id][to.Id].ContainsKey(vehicleToProfileMapper[vehicleProperties]))
00067                 {
00068                     return distanceMatrix[from.Id][to.Id][vehicleToProfileMapper[vehicleProperties]];
00069                 }
00070 
00071             }
00072             return SelfContain ?
00073                 new TimeLengthDistance(from.Id, to.Id, double.MaxValue, double.MaxValue,
      vehicleProperties) :  null;
00074         }
00075     }
00076 }
```

7.5 Driver.cs File Reference

Classes

- class VRPTWOptimizer.Driver

  Represents the truck/tractor driver

Namespaces

- namespace VRPTWOptimizer

7.6 Driver.cs

Go to the documentation of this file.

```
00001 namespace VRPTWOptimizer
00002 {
00006     public abstract class Driver :  Resource
00007     {
00011         public int[] CompatibileVehiclesIds { get; protected set; }
00012 
00020         public Driver(int id, double availabilityStart, double availabilityEnd, int[]
      compatibileVehiclesIds)
00021                     :  base(id, availabilityStart, availabilityEnd)
00022         {
00023             CompatibileVehiclesIds = compatibileVehiclesIds;
00024         }
00025     }
00026 }
```

7.7 PickingSchedule.cs File Reference

Classes

- class VRPTWOptimizer.Dto.PickingSchedule

  Complete picking schedule for one day

Namespaces

- namespace VRPTWOptimizer
- namespace VRPTWOptimizer.Dto

7.8 PickingSchedule.cs

Go to the documentation of this file.

```
00001 using Newtonsoft.Json;
00002 using System;
00003 using System.Collections.Generic;
00004 using System.IO;
00005 using System.Linq;
00006 using System.Text;
00007 using System.Threading.Tasks;
00008 using VRPTWOptimizer.Interfaces;
00009
00010 namespace VRPTWOptimizer.Dto
00011 {
00015     public class PickingSchedule
00016     {
00017         private List<VehicleSchedule> vehicleSchedules = new List<VehicleSchedule>();
00021         [JsonProperty("callbackUrl")]
00022         public string CallbackUrl { get; set; }
00026         [JsonProperty("ordersId")]
00027         public int Id { get; set; }
00031         [JsonProperty("ordersCreateDate")]
00032         public DateTime OrdersCreateDate { get; set; }
00036         [JsonProperty("ordersPickingStart")]
00037         public DateTime OrdersPickingStart { get; set; }
00041         [JsonProperty("vdPickingLists")]
00042         public List<TransportPickingLists> PickingLists { get; set; }
00046         [JsonProperty("vehiclesAvailability")]
00047         public List<VehicleSchedule> VehiclesAvailability { get; set; }
00048
00055         public static PickingSchedule GeneratePickingSchedule(VRPDefinition vrpDefinition, VRPSolution
      vrpSolution)
00056         {
00057             List<TransportPickingLists> transportPickingLists = new List<TransportPickingLists>();
00058             List<VehicleSchedule> vehicleSchedules = new List<VehicleSchedule>();
00059             foreach (var vehicle in vrpDefinition.Vehicles)
00060             {
00061                 var vehicleSchedule = new VehicleSchedule();
00062                 vehicleSchedule.CapacityVehicleType = vehicle.Type;
00063                 vehicleSchedule.VehicleId = vehicle.Id;
00064                 //HACK: this is not right - better to extend the model with informative variables
00065                 vehicleSchedule.EpCapacity = vehicle.RoadProperties.EpCount;
00066                 vehicleSchedule.YardAvailabilitySchedule = new List<TimeInterval>();
00067                 double availabilityStart = vehicle.AvailabilityStart;
00068                 double availabilityEnd = vehicle.AvailabilityEnd;
00069                 foreach (var route in vrpSolution.Transports.Where(tr => tr.TractorId ==
      vehicleSchedule.VehicleId || tr.TrailerTruckId == vehicleSchedule.VehicleId).OrderBy(tr =>
      tr.AvailableForLoadingTime))
00070                 {
00071                     availabilityEnd = route.AvailableForLoadingTime;
00072                     vehicleSchedule.YardAvailabilitySchedule.Add(
00073                         new TimeInterval()
00074                         {
00075                             AvailabilityStart = vrpDefinition.ZeroHour.AddSeconds(availabilityStart),
00076                             AvailabilityEnd = vrpDefinition.ZeroHour.AddSeconds(availabilityEnd)
00077                         });
00078                     availabilityStart = route.AvailableForNextAssignmentTime;
00079                 }
00080                 availabilityEnd = vehicle.AvailabilityEnd;
00081                 vehicleSchedule.YardAvailabilitySchedule.Add(
00082                     new TimeInterval()
00083                     {
00084                         AvailabilityStart = vrpDefinition.ZeroHour.AddSeconds(availabilityStart),
00085                         AvailabilityEnd = vrpDefinition.ZeroHour.AddSeconds(availabilityEnd)
00086                     });
00087                 vehicleSchedules.Add(vehicleSchedule);
00088             }
00089             foreach (var route in vrpSolution.Transports)
00090             {
00091                 Vehicle vehicle = vrpDefinition.Vehicles.First(v => v.Id == route.TrailerTruckId);
00092                 List<StorePickingList> storePickingLists = new List<StorePickingList>();
00093                 var maxAllowedVehicleCapacityForTransport = int.MaxValue;
00094                 foreach (var requestId in route.Schedule[0].LoadedRequestsIds)
00095                 {
00096                     var request = vrpDefinition.Requests.First(rq => rq.Id == requestId);
```

```
00097                     maxAllowedVehicleCapacityForTransport =
      Math.Min(maxAllowedVehicleCapacityForTransport, request.MaxVehicleSize.EpCount);
00098                     var visitsDictionary = route.Schedule
00099                         .Select((sch, idx) => new KeyValuePair<int, string>(idx, sch.LocationId))
00100                         .ToList();
00101
00102                     StorePickingList storePickingList = new StorePickingList()
00103                     {
00104                         DeliveryLocationId = request.DeliveryLocation.Id,
00105                         EpCount = request.Size[0],
00106                         LoadingOrder = visitsDictionary.Count - visitsDictionary.First(vd => vd.Value
      == request.DeliveryLocation.Id).Key - 1,
00107                         GoodsList = new List<CargoUnit>()
00108                     };
00109                     storePickingLists.Add(storePickingList);
00110                 }
00111                 storePickingLists = storePickingLists.OrderBy(st => st.LoadingOrder).ToList();
00112                 for (int i = 0; i < storePickingLists.Count; i++)
00113                 {
00114                     storePickingLists[i].LoadingOrder = i + 1;
00115                 }
00116                 if (storePickingLists.Count > 0)
00117                 {
00118                     var transport = new TransportPickingLists()
00119                     {
00120                         TransportId = route.TransportId,
00121                         CapacityVehicleType = vehicle.Type,
00122                         DesiredDepartureTime =
      vrpDefinition.ZeroHour.AddSeconds(route.Schedule[0].DepartureTime),
00123                         EpCapacity = (int)vehicle.Capacity[0],
00124                         MaxEpCapacity = maxAllowedVehicleCapacityForTransport,
00125                         SemiTrailerTruckId = vehicle.Id,
00126                         StoreOrders = storePickingLists
00127                     };
00128                     transportPickingLists.Add(transport);
00129                 }
00130             }
00131
00132             return new PickingSchedule()
00133             {
00134                 CallbackUrl = "http://fakeurl.com:5050/loadSchedule",
00135                 Id = 1,
00136                 OrdersCreateDate = vrpDefinition.Date.AddDays(-2),
00137                 OrdersPickingStart = vrpDefinition.ZeroHour,
00138                 PickingLists = transportPickingLists,
00139                 VehiclesAvailability = vehicleSchedules
00140             };
00141         }
00142
00149         public static PickingSchedule GeneratePickingSchedule(List<IRoute> routes, List<Vehicle>
      vehicles)
00150         {
00151             throw new NotImplementedException();
00152         }
00153
00158         public void TrySaveToFile(string filename)
00159         {
00160             var settings = new JsonSerializerSettings();
00161             settings.Formatting = Formatting.Indented;
00162             var serializer = JsonSerializer.Create(settings);
00163             var writer = new StringWriter();
00164             serializer.Serialize(writer, this);
00165             try
00166             {
00167                 File.WriteAllText(filename, writer.ToString());
00168             }
00169             catch (IOException)
00170             {
00171                 Console.Write(writer.ToString());
00172             }
00173         }
00174     }
00175 }
```

7.9 StorePickingList.cs File Reference

Classes

- class VRPTWOptimizer.Dto.StorePickingList

  Picking order details for single store

Namespaces

- namespace VRPTWOptimizer
- namespace VRPTWOptimizer.Dto

7.10 StorePickingList.cs

Go to the documentation of this file.

```
00001 using Newtonsoft.Json;
00002 using System;
00003 using System.Collections.Generic;
00004 using System.Linq;
00005 using System.Text;
00006 using System.Threading.Tasks;
00007
00008 namespace VRPTWOptimizer.Dto
00009 {
00013     public class StorePickingList
00014     {
00018         [JsonProperty("storeId")]
00019         public string DeliveryLocationId { get; set; }
00023         [JsonProperty("epCount")]
00024         public double EpCount { get; set; }
00028         [JsonProperty("vdPickingListPosition")]
00029         public List<CargoUnit> GoodsList { get; set; }
00033         [JsonProperty("loadingOrder")]
00034         public int LoadingOrder { get; set; }
00035     }
00036 }
```

7.11 TimeInterval.cs File Reference

Classes

- class VRPTWOptimizer.Dto.TimeInterval

  Represents time interval for vehicle yard availability

Namespaces

- namespace VRPTWOptimizer
- namespace VRPTWOptimizer.Dto

7.12 TimeInterval.cs

Go to the documentation of this file.

```
00001 using Newtonsoft.Json;
00002 using System;
00003 using System.Collections.Generic;
00004 using System.Linq;
00005 using System.Text;
00006 using System.Threading.Tasks;
00007
00008 namespace VRPTWOptimizer.Dto
00009 {
00013     public class TimeInterval
00014     {
00018         [JsonProperty("yardAvailabilityEnd")]
00019         public DateTime AvailabilityEnd { get; set; }
00023         [JsonProperty("yardAvailabilityStart")]
00024         public DateTime AvailabilityStart { get; set; }
00025     }
00026 }
```

7.13 TransportPickingLists.cs File Reference

Classes

- class VRPTWOptimizer.Dto.TransportPickingLists

  Picking orders for single transport

Namespaces

- namespace VRPTWOptimizer
- namespace VRPTWOptimizer.Dto

7.14 TransportPickingLists.cs

Go to the documentation of this file.

```
00001 using CommonGIS.Enums;
00002 using Newtonsoft.Json;
00003 using System;
00004 using System.Collections.Generic;
00005 using System.Linq;
00006 using System.Text;
00007 using System.Threading.Tasks;
00008
00009 namespace VRPTWOptimizer.Dto
00010 {
00014     public class TransportPickingLists
00015     {
00019         [JsonProperty("capacityVehicleType")]
00020         public VehicleType CapacityVehicleType { get; set; }
00021         //TODO MO drukować datę bez strefy czasowej
00025         [JsonProperty("desiredDepartureTime")]
00026         public DateTime DesiredDepartureTime { get; set; }
00030         [JsonProperty("epCapacity")]
00031         public int EpCapacity { get; set; }
00035         [JsonProperty("maxEpCapacity")]
00036         public int MaxEpCapacity { get; set; }
00040         [JsonProperty("semiTrailerTruckId")]
00041         public int SemiTrailerTruckId { get; set; }
00045         [JsonProperty("vdPickingListStores")]
00046         public List<StorePickingList> StoreOrders { get; set; }
00050         [JsonProperty("transportId")]
00051         public int TransportId { get; set; }
00052     }
00053 }
```

7.15 VehicleSchedule.cs File Reference

Classes

- class VRPTWOptimizer.Dto.VehicleSchedule

  Schedules vehicle presence at warehouse grounds

Namespaces

- namespace VRPTWOptimizer
- namespace VRPTWOptimizer.Dto

7.16 VehicleSchedule.cs

Go to the documentation of this file.

```
00001 using CommonGIS.Enums;
00002 using Newtonsoft.Json;
00003 using System;
00004 using System.Collections.Generic;
00005 using System.Linq;
00006 using System.Text;
00007 using System.Threading.Tasks;
00008
00009 namespace VRPTWOptimizer.Dto
00010 {
00014     public class VehicleSchedule
00015     {
00019         [JsonProperty("capacityVehicleType")]
00020         public VehicleType CapacityVehicleType { get; set; }
00024         [JsonProperty("epCapacity")]
00025         public int EpCapacity { get; set; }
00029         [JsonProperty("semiTrailerTruckId")]
00030         public int VehicleId { get; set; }
00034         [JsonProperty("schedule")]
00035         public List<TimeInterval> YardAvailabilitySchedule { get; set; }
00036     }
00037 }
```

7.17 Aggregation.cs File Reference

Namespaces

- namespace VRPTWOptimizer
- namespace VRPTWOptimizer.Enums

Enumerations

- enum VRPTWOptimizer.Enums.Aggregation { VRPTWOptimizer.Enums.Sum = 1 , VRPTWOptimizer.Enums.Max = 2 }

  Enumerator describing available types of cargo size aggregation

7.18 Aggregation.cs

Go to the documentation of this file.

```
00001 namespace VRPTWOptimizer.Enums
00002 {
00006     public enum Aggregation
00007     {
00011         Sum = 1,
00015         Max = 2
00016     }
00017 }
```

7.19 CargoType.cs File Reference

Namespaces

- namespace VRPTWOptimizer
- namespace VRPTWOptimizer.Enums

Enumerations

- enum VRPTWOptimizer.Enums.CargoType { VRPTWOptimizer.Enums.Food = 1 , VRPTWOptimizer.Enums.EmptyBoxes = 2 , VRPTWOptimizer.Enums.Garbage = 3 }

  Type of cargo that would determine if it can be transported with other types

7.20 CargoType.cs

Go to the documentation of this file.

```
00001 namespace VRPTWOptimizer.Enums
00002 {
00006     public enum CargoType
00007     {
00011         Food = 1,
00015         EmptyBoxes = 2,
00019         Garbage = 3,
00020     }
00021 }
```

7.21 CargoUnitType.cs File Reference

Namespaces

- namespace VRPTWOptimizer
- namespace VRPTWOptimizer.Enums

Enumerations

- enum VRPTWOptimizer.Enums.CargoUnitType { VRPTWOptimizer.Enums.Box = 1 }

  Types of cargo units

7.22 CargoUnitType.cs

Go to the documentation of this file.

```
00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using System.Text;
00005 using System.Threading.Tasks;
00006
00007 namespace VRPTWOptimizer.Enums
00008 {
00012     public enum CargoUnitType
00013     {
00017         Box = 1
00018     }
00019 }
```

7.23 RequestType.cs File Reference

Namespaces

- namespace VRPTWOptimizer
- namespace VRPTWOptimizer.Enums

Enumerations

- enum VRPTWOptimizer.Enums.RequestType { VRPTWOptimizer.Enums.GoodsDistribution = 1 , VRPTWOptimizer.Enums.Con = 2 , VRPTWOptimizer.Enums.Backhauling = 3 }

  Type of TransportRequest possibly useful to set priorities

7.24 RequestType.cs

Go to the documentation of this file.

```
00001 namespace VRPTWOptimizer.Enums
00002 {
00006     public enum RequestType
00007     {
00011         GoodsDistribution = 1,
00015         ContainerRetrieval = 2,
00019         Backhauling = 3,
00020     }
00021 }
```

7.25 InsertionResult.cs File Reference

Classes

- class VRPTWOptimizer.InsertionResult

  Description of TransportRequest insert consequences

Namespaces

- namespace VRPTWOptimizer

7.26 InsertionResult.cs

Go to the documentation of this file.

```
00001 namespace VRPTWOptimizer
00002 {
00006     public class InsertionResult
00007     {
00011         public double ExpectedArriveTime { get; }
00015         public double MaxDelay { get; }
00019         public double NewAddedDistance { get; }
00020         public double NewNextArriveTime { get; }
00021
00022         public double OldDistanceBetween { get; }
00023
00024         public double OldNextArriveTime { get; }
00025
00026         public bool Success { get; }
00027
00028         public InsertionResult(
00029             double oldDistanceBetween,
00030             double newAddedDistance,
00031             double oldNextArriveTime,
00032             double newNextArriveTime,
00033             double expectedArriveTime,
00034             double maxDelay,
00035             bool success)
00036         {
00037             OldDistanceBetween = oldDistanceBetween;
00038             OldNextArriveTime = oldNextArriveTime;
00039             NewNextArriveTime = newNextArriveTime;
00040             ExpectedArriveTime = expectedArriveTime;
00041             Success = success;
00042             NewAddedDistance = newAddedDistance;
00043             MaxDelay = maxDelay;
00044         }
00045     }
00046 }
```

7.27 IRoute.cs File Reference

Classes

- interface VRPTWOptimizer.Interfaces.IRoute

  Represents route assigned to Vehicle

Namespaces

- namespace VRPTWOptimizer
- namespace VRPTWOptimizer.Interfaces

7.28 IRoute.cs

Go to the documentation of this file.

```
00001 using CommonGIS;
00002 using System.Collections.Generic;
00003
00004 namespace VRPTWOptimizer.Interfaces
00005 {
00009     public interface IRoute
00010     {
00014         List<double> ArrivalTimes { get; }
00018         List<double> DepartureTimes { get; }
00022         List<Distance> Distances { get; }
00026         double Length { get; }
00030         List<List<TransportRequest» LoadedRequests { get; }
00034         double MaxDelay { get; }
00038         List<double> TimeWindowEnd { get; }
00042         List<double> TimeWindowStart { get; }
00046         double TotalDelay { get; }
00050         double TravelTime { get; }
00054         List<List<TransportRequest» UnloadedRequests { get; }
00058         Vehicle Vehicle { get; }
00062         Driver VehicleDriver { get; }
00066         Vehicle VehicleTractor { get; }
00070         List<Location> VisitedLocations { get; }
00071     }
00072 }
```

7.29 ITimeEstimator.cs File Reference

Classes

- interface VRPTWOptimizer.Interfaces.ITimeEstimator

  Predicts time of loading and unloading cargo at Location

Namespaces

- namespace VRPTWOptimizer
- namespace VRPTWOptimizer.Interfaces

7.30 ITimeEstimator.cs

Go to the documentation of this file.

```
00001 namespace VRPTWOptimizer.Interfaces
00002 {
00006     public interface ITimeEstimator
00007     {
00016         double EstimateLoadUnloadTime(int epUnloadCount, int epLoadOnlyCount, int
      epImmediatelyRetrievedCount, int handledTransportRequestsCount);
00017     }
00018 }
```

7.31 IVRPOptimizer.cs File Reference

Classes

- interface VRPTWOptimizer.Interfaces.IVRPOptimizer

  Optimizes Vehicle Routing Problem

Namespaces

- namespace VRPTWOptimizer
- namespace VRPTWOptimizer.Interfaces

7.32 IVRPOptimizer.cs

Go to the documentation of this file.

```
00001 using CommonGIS.Interfaces;
00002 using System;
00003
00004 namespace VRPTWOptimizer.Interfaces
00005 {
00009     public interface IVRPOptimizer
00010     {
00018         [Obsolete("Please use Optimize with conscious costFunctionFactors definition")]
00019         VRPOptimizerResult Optimize(
00020            IVRPProvider problemDataProvider,
00021            ITimeEstimator serviceTimeEstimator,
00022            IDistanceProvider distanceProvider);
00023
00032         VRPOptimizerResult Optimize(
00033            IVRPProvider problemDataProvider,
00034            VRPCostFunction costFunctionFactors,
00035            ITimeEstimator serviceTimeEstimator,
00036            IDistanceProvider distanceProvider);
00037     }
00038 }
```

7.33 IVRPOptimizerFactory.cs File Reference

Classes

- interface VRPTWOptimizer.Interfaces.IVRPOptimizerFactory

  Provides new instance of optimizer (follows Abstract Factory desing pattern)

Namespaces

- namespace VRPTWOptimizer
- namespace VRPTWOptimizer.Interfaces

7.34 IVRPOptimizerFactory.cs

Go to the documentation of this file.

```
00001 using System.Collections.Generic;
00002
00003 namespace VRPTWOptimizer.Interfaces
00004 {
00008     public interface IVRPOptimizerFactory
00009     {
00014         IVRPOptimizer CreateOptimizer();
00015
00020         IVRPOptimizer CreateOptimizer(Dictionary<string, object> configuration);
00021     }
00022 }
```

7.35 IVRPProvider.cs File Reference

Classes

- interface VRPTWOptimizer.Interfaces.IVRPProvider

  Provides problem data for IVRPOptimizer

Namespaces

- namespace VRPTWOptimizer
- namespace VRPTWOptimizer.Interfaces

7.36 IVRPProvider.cs

Go to the documentation of this file.

```
00001 using CommonGIS;
00002 using System;
00003 using System.Collections.Generic;
00004
00005 namespace VRPTWOptimizer.Interfaces
00006 {
00010     public interface IVRPProvider
00011     {
00015         List<Driver> Drivers { get; }
00019         Location HomeDepot { get; }
00023         List<TransportRequest> Requests { get; }
00027         List<Vehicle> Vehicles { get; }
00031         DateTime ZeroHour { get; }
00032
00038         void LoadData(DateTime billingDate, string homeDepotId);
00039     }
00040 }
```

7.37 IVRPSolutionWriter.cs File Reference

Classes

- interface VRPTWOptimizer.Interfaces.IVRPSolutionWriter

Namespaces

- namespace VRPTWOptimizer
- namespace VRPTWOptimizer.Interfaces

7.38 IVRPSolutionWriter.cs

Go to the documentation of this file.

```
00001 using System;
00002 using System.Collections.Generic;
00003
00004 namespace VRPTWOptimizer.Interfaces
00005 {
00006     public interface IVRPSolutionWriter
00007     {
00008         public void SaveSolution(List<IRoute> routes, List<TransportRequest> unassignedRequests,
      DateTime billingDate, string homeDepotId, string algorithmName);
00009     }
00010 }
```

7.39 VRPResult.cs File Reference

Classes

- class VRPTWOptimizer.Interfaces.VRPResult< R, V, Rt >

  Class containing list of routes for each tractor and straight truck

Namespaces

- namespace VRPTWOptimizer
- namespace VRPTWOptimizer.Interfaces

7.40 VRPResult.cs

Go to the documentation of this file.

```
00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004
00005 namespace VRPTWOptimizer.Interfaces
00006 {
00013     public class VRPResult<R, V, Rt>
00014         where R : TransportRequest
00015         where V : Vehicle
00016         where Rt :  IRoute
00017     {
00021         public List<R> LeftRequests { get; }
00022
00026         [Obsolete]
00027         public List<Rt> Routes => TractorRoutes.SelectMany(r => r.Value).ToList();
00031         public Dictionary<V, List<Rt>> TractorRoutes { get; }
00032
00038         public VRPResult(Dictionary<V, List<Rt>> routes, List<R> leftRequests)
00039         {
00040             LeftRequests = leftRequests;
00041             TractorRoutes = routes;
00042         }
00043
00048         [Obsolete]
00049         public static implicit operator Tuple<List<Rt>, List<R>>(VRPResult<R, V, Rt> vrp) => new
      Tuple<List<Rt>, List<R>>(vrp.Routes, vrp.LeftRequests);
00050     }
00051 }
```

7.41 JSONDefinitionWriter.cs File Reference

Classes

- class VRPTWOptimizer.Logging.JSONDefinitionWriter

  Serializes solution to JSON

Namespaces

- namespace VRPTWOptimizer
- namespace VRPTWOptimizer.Logging

7.42 JSONDefinitionWriter.cs

Go to the documentation of this file.

```
00001 using CommonGIS.Interfaces;
00002 using System;
00003 using System.Collections.Generic;
00004 using System.IO;
00005 using System.Linq;
00006 using System.Reflection;
00007 using VRPTWOptimizer.Interfaces;
00008
00009 namespace VRPTWOptimizer.Logging
00010 {
00014     public class JSONDefinitionWriter :  IVRPSolutionWriter
00015
00016     {
00017         private readonly string _clientName;
00018         private readonly DateTime _computationsEnd;
00019         private readonly DateTime _computationsStart;
00020         private readonly IDistanceProvider _distanceData;
00021         private readonly string _filename;
00022         private readonly IVRPOptimizer _optimizer;
00023         private readonly List<TransportRequest> _requests;
00024         private readonly ITimeEstimator _timeEstimator;
00025         private readonly List<Vehicle> _vehicles;
00026         private readonly DateTime _zeroHour;
00027
00028         [Obsolete("Serialization of result should be done with serializing VRPDefinition with
      VRPSolutions collection")]
00029         public JSONDefinitionWriter(List<TransportRequest> requests,
00030                                     List<Vehicle> vehicles,
00031                                     IVRPOptimizer optimizer,
00032                                     DateTime zeroHour,
00033                                     DateTime computationsEnd,
00034                                     DateTime computationsStart,
00035                                     IDistanceProvider distanceData,
00036                                     ITimeEstimator timeEstimator,
00037                                     string filename,
00038                                     string clientName)
00039         {
00040             _requests = requests;
00041             _vehicles = vehicles;
00042             _optimizer = optimizer;
00043             _zeroHour = zeroHour;
00044             _computationsEnd = computationsEnd;
00045             _computationsStart = computationsStart;
00046             _distanceData = distanceData;
00047             _timeEstimator = timeEstimator;
00048             _filename = filename;
00049             _clientName = clientName;
00050         }
00051
00060         public void SaveSolution(List<IRoute> routes, List<TransportRequest> unassignedRequests,
      DateTime billingDate, string homeDepotId, string algorithmName)
00061         {
00062             var vrpDefinition = new VRPDefinition();
00063             vrpDefinition.Requests = _requests;
00064             vrpDefinition.ServiceTimeEstimator = _timeEstimator;
00065             vrpDefinition.Vehicles = _vehicles;
00066             var vrpSolution = new VRPSolution();
```

```
00067             vrpDefinition.Solutions = new List<VRPSolution>();
00068             vrpDefinition.Date = billingDate.Date;
00069             vrpDefinition.DepotId = homeDepotId;
00070             vrpDefinition.Client = _clientName;
00071             vrpDefinition.ZeroHour = _zeroHour;
00072             vrpDefinition.Solutions.Add(vrpSolution);
00073             vrpDefinition.DistanceData = _distanceData;
00074 
00075             vrpSolution.LeftRequestsIds = unassignedRequests.Select(req => req.Id).ToList();
00076             vrpSolution.ComputationTimestamp = _computationsEnd;
00077             vrpSolution.ComputationTime = (_computationsEnd - _computationsStart).TotalSeconds;
00078             vrpSolution.ComputerId = Environment.MachineName;
00079             vrpSolution.Version = Assembly.GetAssembly(_optimizer.GetType()).GetName().Version;
00080             var orderedTrailerAssignment = routes
00081                 .OrderBy(rt => rt.ArrivalTimes[0]);
00082             int transportId = 1;
00083             vrpSolution.DelaysCount = routes.Count(rt => rt.TotalDelay >= 0);
00084             vrpSolution.MaxDelay = routes.Max(rt => rt.MaxDelay);
00085             vrpSolution.TotalDelay = routes.Sum(rt => rt.TotalDelay);
00086             vrpSolution.TotalLength = routes.Sum(rt => rt.Length);
00087             vrpSolution.Transports = new List<VRPSolution.TransportItem>();
00088             vrpSolution.Algorithm = _optimizer.GetType().FullName;
00089             foreach (var assignment in orderedTrailerAssignment)
00090             {
00091                 double fillInRatio = assignment.Vehicle.Capacity
00092                         .Select((capacity, index) => index)
00093                         .Max(index => assignment.LoadedRequests[0].Sum(rq => rq.Size[index]) /
    assignment.Vehicle.Capacity[index])
00094                         ;
00095                 var transport = new VRPSolution.TransportItem();
00096                 transport.TransportId = transportId;
00097                 transport.TractorId = -1;
00098                 transport.TrailerTruckId = assignment.Vehicle.Id;
00099                 transport.Length = assignment.Length;
00100                 transport.Schedule = new List<VRPSolution.ScheduleItem>();
00101                 transport.AvailableForLoadingTime = assignment.ArrivalTimes[0];
00102                 transport.AvailableForNextAssignmentTime =
    assignment.DepartureTimes[assignment.DepartureTimes.Count - 1];
00103                 transport.FillInRatio = Math.Round(fillInRatio, 2);
00104                 for (int i = 0; i < assignment.VisitedLocations.Count; i++)
00105                 {
00106                     var scheduleItem = new VRPSolution.ScheduleItem();
00107                     scheduleItem.LocationId = assignment.VisitedLocations[i].Id;
00108                     scheduleItem.ArrivalTime = assignment.ArrivalTimes[i];
00109                     scheduleItem.DepartureTime = assignment.DepartureTimes[i];
00110                     scheduleItem.Delay = Math.Max(assignment.ArrivalTimes[i] -
    assignment.TimeWindowEnd[i], 0);
00111                     scheduleItem.LoadedRequestsIds = assignment.LoadedRequests[i].Select(rq =>
    rq.Id).ToList();
00112                     scheduleItem.UnloadedRequestsIds = assignment.UnloadedRequests[i].Select(rq =>
    rq.Id).ToList();
00113                     transport.Schedule.Add(scheduleItem);
00114                 }
00115                 vrpSolution.Transports.Add(transport);
00116                 transportId++;
00117             }
00118 
00119             File.WriteAllText(_filename, vrpDefinition.ToPrettyJSONString());
00120         }
00121     }
00122 }
```

7.43 .NETCoreApp,Version=v5.0.AssemblyAttributes.cs File Reference

7.44 Debug/net5.0/.NETCoreApp,Version=v5.0.AssemblyAttributes.cs

Go to the documentation of this file.

```
00001 // <autogenerated />
00002 using System;
00003 using System.Reflection;
00004 [assembly:  global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v5.0",
    FrameworkDisplayName = ".NET 5.0")]
```

7.45 .NETCoreApp,Version=v5.0.AssemblyAttributes.cs File Reference

7.46 Release/net5.0/.NETCoreApp,Version=v5.0.AssemblyAttributes.cs

Go to the documentation of this file.

```
00001 // <autogenerated />
00002 using System;
00003 using System.Reflection;
00004 [assembly:  global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v5.0",
      FrameworkDisplayName = ".NET 5.0")]
```

7.47 VRPTWOptimizer.AssemblyInfo.cs File Reference

7.48 Debug/net5.0/VRPTWOptimizer.AssemblyInfo.cs

Go to the documentation of this file.

```
00001 //------------------------------------------------------------------------------
00002 // <auto-generated>
00003 //     This code was generated by a tool.
00004 //     Runtime Version:4.0.30319.42000
00005 //
00006 //     Changes to this file may cause incorrect behavior and will be lost if
00007 //     the code is regenerated.
00008 // </auto-generated>
00009 //------------------------------------------------------------------------------
00010
00011 using System;
00012 using System.Reflection;
00013
00014 [assembly:  System.Reflection.AssemblyCompanyAttribute("VRPTWOptimizer")]
00015 [assembly:  System.Reflection.AssemblyConfigurationAttribute("Debug")]
00016 [assembly:  System.Reflection.AssemblyDescriptionAttribute("Library with basic Vehicle Routing Problem
      data types and interfaces")]
00017 [assembly:  System.Reflection.AssemblyFileVersionAttribute("4.0.1.0")]
00018 [assembly:
      System.Reflection.AssemblyInformationalVersionAttribute("4.0.1+9f5cb7f9e9da0f78db7c2d604e6bb3087e570641")]
00019 [assembly:  System.Reflection.AssemblyProductAttribute("VRPTWOptimizer")]
00020 [assembly:  System.Reflection.AssemblyTitleAttribute("VRPTWOptimizer")]
00021 [assembly:  System.Reflection.AssemblyVersionAttribute("4.0.1.0")]
00022 [assembly:  System.Reflection.AssemblyMetadataAttribute("RepositoryUrl",
      "https://bitbucket.org/control-system/autonomicznyspedytor.git")]
00023
00024 // Generated by the MSBuild WriteCodeFragment class.
00025
```

7.49 VRPTWOptimizer.AssemblyInfo.cs File Reference

7.50 Release/net5.0/VRPTWOptimizer.AssemblyInfo.cs

Go to the documentation of this file.

```
00001 //------------------------------------------------------------------------------
00002 // <auto-generated>
00003 //     This code was generated by a tool.
00004 //     Runtime Version:4.0.30319.42000
00005 //
00006 //     Changes to this file may cause incorrect behavior and will be lost if
00007 //     the code is regenerated.
00008 // </auto-generated>
00009 //------------------------------------------------------------------------------
00010
00011 using System;
00012 using System.Reflection;
00013
00014 [assembly:  System.Reflection.AssemblyCompanyAttribute("VRPTWOptimizer")]
```

```
00015 [assembly:  System.Reflection.AssemblyConfigurationAttribute("Release")]
00016 [assembly:  System.Reflection.AssemblyDescriptionAttribute("Library with basic Vehicle Routing Problem
      data types and interfaces")]
00017 [assembly:  System.Reflection.AssemblyFileVersionAttribute("4.0.1.0")]
00018 [assembly:  System.Reflection.AssemblyInformationalVersionAttribute("4.0.1")]
00019 [assembly:  System.Reflection.AssemblyProductAttribute("VRPTWOptimizer")]
00020 [assembly:  System.Reflection.AssemblyTitleAttribute("VRPTWOptimizer")]
00021 [assembly:  System.Reflection.AssemblyVersionAttribute("4.0.1.0")]
00022 [assembly:  System.Reflection.AssemblyMetadataAttribute("RepositoryUrl",
      "https://bitbucket.org/control-system/autonomicznyspedytor.git")]
00023
00024 // Generated by the MSBuild WriteCodeFragment class.
00025
```

7.51 Resource.cs File Reference

Classes

- class VRPTWOptimizer.Resource

  Generalized time bound resource (driver, machine, vehicle)

Namespaces

- namespace VRPTWOptimizer

7.52 Resource.cs

Go to the documentation of this file.

```
00001 using System;
00002
00003 namespace VRPTWOptimizer
00004 {
00008     public class Resource
00009     {
00010         private const string FINITE_AVAILABILITY_ERROR = "Vehicle must have finite availability";
00014         public double AvailabilityEnd { get; protected set; }
00015
00019         public double AvailabilityStart { get; protected set; }
00020
00024         public int Id { get; protected set; }
00025
00032         public Resource(int id, double availabilityStart, double availabilityEnd)
00033         {
00034             AvailabilityEnd = availabilityEnd;
00035             AvailabilityStart = availabilityStart;
00036             Id = id;
00037             if (!double.IsFinite(AvailabilityStart))
00038                 throw new ArgumentException(FINITE_AVAILABILITY_ERROR);
00039             if (!double.IsFinite(AvailabilityEnd))
00040                 throw new ArgumentException(FINITE_AVAILABILITY_ERROR);
00041         }
00042     }
00043 }
```

7.53 TransportRequest.cs File Reference

Classes

- class VRPTWOptimizer.TransportRequest

  Description of the request to move cargo from pickup to delivery Location

Namespaces

- namespace VRPTWOptimizer

7.54 TransportRequest.cs

Go to the documentation of this file.

```
00001 using CommonGIS;
00002 using CommonGIS.Enums;
00003 using CommonGIS.Interfaces;
00004 using System;
00005 using System.Collections.Generic;
00006 using System.Linq;
00007 using VRPTWOptimizer.Enums;
00008 using VRPTWOptimizer.Interfaces;
00009
00010 namespace VRPTWOptimizer
00011 {
00015     public abstract class TransportRequest
00016     {
00020         public int CargoType => CargoTypes.FirstOrDefault();
00024         public int[] CargoTypes { get; protected set; }
00028         public double DeliveryAvailableTimeWindowEnd { get; protected set; }
00032         public double DeliveryAvailableTimeWindowStart { get; protected set; }
00036         public Location DeliveryLocation { get; protected set; }
00037
00041         public double DeliveryPreferedTimeWindowEnd { get; protected set; }
00042
00046         public double DeliveryPreferedTimeWindowStart { get; protected set; }
00050         public int Id { get; protected set; }
00054         public VehicleRoadRestrictionProperties MaxVehicleSize { get; protected set; }
00059         public Dictionary<int, double> MutuallyExclusiveRequestsIdTimeBufferDict { get; protected set;
    }
00063         public string Name { get; protected set; }
00068         public int[] NecessaryVehicleSpecialProperties { get; protected set; }
00073         public int PackageCount { get; protected set; }
00074
00078         public int PackageCountForImediateRetrieval { get; protected set; }
00079
00083         public double PickupAvailableTimeWindowEnd { get; protected set; }
00084
00088         public double PickupAvailableTimeWindowStart { get; protected set; }
00089
00093         public Location PickupLocation { get; protected set; }
00094
00098         public double PickupPreferedTimeWindowEnd { get; protected set; }
00099
00103         public double PickupPreferedTimeWindowStart { get; protected set; }
00104
00108         public int[] RestrictedCargoTypes { get; protected set; }
00109
00113         public double RevenueValue { get; protected set; }
00114
00119         public double[] Size { get; protected set; }
00123         public RequestType Type { get; protected set; }
00124
00150         public TransportRequest(int id,
00151                                 double[] size,
00152                                 int[] necessaryVehicleSpecialProperties,
00153                                 int packageCount,
00154                                 int packageCountForImediateRetrieval,
00155                                 Location startLocation,
00156                                 double pickupAvailableTimeWindowStart,
00157                                 double pickupPreferedTimeWindowStart,
00158                                 double pickupPreferedTimeWindowEnd,
00159                                 double pickupAvailableTimeWindowEnd,
00160                                 Location endLocation,
00161                                 double deliveryAvailableTimeWindowStart,
00162                                 double deliveryPreferedTimeWindowStart,
00163                                 double deliveryPreferedTimeWindowEnd,
00164                                 double deliveryAvailableTimeWindowEnd,
00165                                 RequestType requestType,
00166                                 int[] cargoTypes,
00167                                 VehicleRoadRestrictionProperties maxVehicleSize,
00168                                 int[] restrictedGoodsTypes,
00169                                 Dictionary<int, double> mutuallyExclusiveRequestsIdTimeBufferDict,
00170                                 double revenueValue,
00171                                 string name)
00172         {
00173             Size = new double[size.Length];
```

```
00174             Array.Copy(size, Size, size.Length);
00175             CargoTypes = new int[cargoTypes.Length];
00176             Array.Copy(cargoTypes, CargoTypes, cargoTypes.Length);
00177 
00178             RestrictedCargoTypes = new int[restrictedGoodsTypes.Length];
00179             Array.Copy(restrictedGoodsTypes, restrictedGoodsTypes, restrictedGoodsTypes.Length);
00180             PackageCount = packageCount;
00181             PickupLocation = startLocation;
00182             DeliveryLocation = endLocation;
00183             DeliveryPreferedTimeWindowStart = deliveryPreferedTimeWindowStart;
00184             DeliveryPreferedTimeWindowEnd = deliveryPreferedTimeWindowEnd;
00185             MaxVehicleSize = maxVehicleSize;
00186             Type = requestType;
00187             PackageCountForImediateRetrieval = packageCountForImediateRetrieval;
00188             Name = name;
00189             Id = id;
00190             MutuallyExclusiveRequestsIdTimeBufferDict = mutuallyExclusiveRequestsIdTimeBufferDict;
00191             DeliveryAvailableTimeWindowEnd = deliveryAvailableTimeWindowEnd;
00192             DeliveryAvailableTimeWindowStart = deliveryAvailableTimeWindowStart;
00193             PickupAvailableTimeWindowEnd = pickupAvailableTimeWindowEnd;
00194             PickupAvailableTimeWindowStart = pickupAvailableTimeWindowStart;
00195             PickupPreferedTimeWindowEnd = pickupPreferedTimeWindowEnd;
00196             PickupPreferedTimeWindowStart = pickupPreferedTimeWindowStart;
00197             RevenueValue = revenueValue;
00198             NecessaryVehicleSpecialProperties = necessaryVehicleSpecialProperties;
00199         }
00200 
00201         internal bool IsArrivalFeasible(double arrivalTimeAtRequest)
00202         {
00203             return arrivalTimeAtRequest >= this.DeliveryPreferedTimeWindowStart &&
    arrivalTimeAtRequest <= this.DeliveryPreferedTimeWindowEnd;
00204         }
00205 
00213         public List<TransportRequest> ExtractBestFitRequests(List<TransportRequest> requests,
    ITimeEstimator timeEstimator, IDistanceProvider distanceProvider)
00214         {
00215             VehicleRoadRestrictionProperties defaultVehicleProperties = new
    VehicleRoadRestrictionProperties(
00216                 VehicleRoadRestrictionProperties.MaxGrossVehicleWeight, 0, 0, 0,
    VehicleTypeRouting.TractorWithTrailer);
00217             var bestFitRequests = new List<TransportRequest>();
00218             double bestbackhaulingDistance = double.PositiveInfinity;
00219             foreach (TransportRequest request in requests)
00220             {
00221                 if (this.Type == RequestType.Backhauling && request.Type ==
    RequestType.GoodsDistribution)
00222                 {
00223                     Distance distanceBetween = distanceProvider
00224                         .GetDistance(request.DeliveryLocation, this.PickupLocation,
    defaultVehicleProperties);
00225                     Distance backhaulingDistance = distanceProvider
00226                         .GetDistance(this.PickupLocation, this.DeliveryLocation,
    defaultVehicleProperties);
00227                     double unloadOldGoodsServiceTime =
    timeEstimator.EstimateLoadUnloadTime(request.PackageCount, 0, 0, 1);
00228                     double loadNewGoodsServiceTime = timeEstimator.EstimateLoadUnloadTime(0,
    this.PackageCount, 0, 1);
00229                     double arrivalTime = request.DeliveryPreferedTimeWindowStart +
    unloadOldGoodsServiceTime + distanceBetween.Time + loadNewGoodsServiceTime + backhaulingDistance.Time;
00230                     if (arrivalTime < this.DeliveryPreferedTimeWindowEnd &&
    request.MaxVehicleSize.DoesVehicleFitIntoRestrictions(this.MaxVehicleSize) &&
    this.MaxVehicleSize.DoesVehicleFitIntoRestrictions(request.MaxVehicleSize))
00231                     {
00232                         if (distanceBetween.Length < bestbackhaulingDistance)
00233                         {
00234                             bestFitRequests.Clear();
00235                             bestbackhaulingDistance = distanceBetween.Length;
00236                             bestFitRequests.Add(request);
00237                         }
00238                     }
00239                 }
00240                 else if (this.Type == RequestType.GoodsDistribution && request.Type ==
    RequestType.ContainerRetrieval)
00241                 {
00242                     Distance distance = distanceProvider
00243                         .GetDistance(this.DeliveryLocation, request.PickupLocation,
    defaultVehicleProperties);
00244                     if (distance.Length == 0)
00245                     {
00246                         if (this.DeliveryPreferedTimeWindowStart <=
    request.PickupPreferedTimeWindowEnd)
00247                         {
00248                             bestFitRequests.Add(request);
00249                         }
00250                     }
00251                 }
00252             }
```

```
00253             return bestFitRequests;
00254         }
00255     }
00256 }
```

7.55 Vehicle.cs File Reference

Classes

- class VRPTWOptimizer.Vehicle

  Defines properties of a Vehicle

Namespaces

- namespace VRPTWOptimizer

7.56 Vehicle.cs

Go to the documentation of this file.

```
00001 using CommonGIS;
00002 using CommonGIS.Enums;
00003 using System;
00004 using System.Collections.Generic;
00005 using System.Linq;
00006 using VRPTWOptimizer.Enums;
00007
00008 namespace VRPTWOptimizer
00009 {
00013     public abstract class Vehicle :  Resource
00014     {
00019         public double[] Capacity { get; protected set; }
00023         public Aggregation[] CapacityAggregationType { get; protected set; }
00027         public Location FinalLocation { get; protected set; }
00031         public Location InitialLocation { get; protected set; }
00035         public double MaxRideTime { get; protected set; }
00039         public int OwnerID { get; protected set; }
00043         public VehicleRoadRestrictionProperties RoadProperties { get; protected set; }
00047         public int[] SpecialProperties { get; protected set; }
00051         public VehicleType Type { get; protected set; }
00055         public double VehicleCostPerDistanceUnit { get; protected set; }
00059         public double VehicleCostPerTimeUnit { get; protected set; }
00063         public double VehicleCostPerUsage { get; protected set; }
00064         //TODO 1 MO add this to constructor!
00065         //TODO MO Wykorzystać te informacje
00069         public double VehicleFlatCostForShortRouteLength { get; }
00074         public double VehicleMaxRouteLengthForFlatCost { get; }
00075
00094         public Vehicle(int id,
00095                        double[] capacity,
00096                        int[] specialProperties,
00097                        Aggregation[] capacityAggregation,
00098                        Location initialLocation,
00099                        double availabilityStart,
00100                        Location finalLocation,
00101                        double availabilityEnd,
00102                        double maxRideTime,
00103                        VehicleRoadRestrictionProperties roadProperties,
00104                        VehicleType type,
00105                        double vehicleCostPerDistanceUnit,
00106                        double vehicleCostPerTimeUnit,
00107                        double vehicleCostPerUsage,
00108                        int ownerID) : this(id,
00109                                            capacity,
00110                                            specialProperties,
00111                                            capacityAggregation,
00112                                            initialLocation,
00113                                            availabilityStart,
00114                                            finalLocation,
00115                                            availabilityEnd,
```

```
00116                                           maxRideTime,
00117                                           roadProperties,
00118                                           type,
00119                                           vehicleCostPerDistanceUnit,
00120                                           vehicleCostPerTimeUnit,
00121                                           vehicleCostPerUsage,
00122                                           ownerID,
00123                                           vehicleFlatCostForShortRouteLength:  0,
00124                                           vehicleMaxRouteLengthForFlatCost:  0)
00125         {
00126         }
00127
00148         public Vehicle(int id,
00149                double[] capacity,
00150                int[] specialProperties,
00151                Aggregation[] capacityAggregation,
00152                Location initialLocation,
00153                double availabilityStart,
00154                Location finalLocation,
00155                double availabilityEnd,
00156                double maxRideTime,
00157                VehicleRoadRestrictionProperties roadProperties,
00158                VehicleType type,
00159                double vehicleCostPerDistanceUnit,
00160                double vehicleCostPerTimeUnit,
00161                double vehicleCostPerUsage,
00162                int ownerID,
00163                double vehicleFlatCostForShortRouteLength,
00164                double vehicleMaxRouteLengthForFlatCost) :  base(id, availabilityStart,
      availabilityEnd)
00165         {
00166             Capacity = new double[capacity.Length];
00167             Array.Copy(capacity, Capacity, capacity.Length);
00168             CapacityAggregationType = new Aggregation[capacityAggregation.Length];
00169             Array.Copy(capacityAggregation, CapacityAggregationType, capacityAggregation.Length);
00170             SpecialProperties = new int[specialProperties.Length];
00171             Array.Copy(specialProperties, SpecialProperties, specialProperties.Length);
00172             InitialLocation = initialLocation;
00173             FinalLocation = finalLocation;
00174             MaxRideTime = maxRideTime;
00175             Type = type;
00176             RoadProperties = roadProperties;
00177             VehicleCostPerDistanceUnit = vehicleCostPerDistanceUnit;
00178             VehicleCostPerTimeUnit = vehicleCostPerTimeUnit;
00179             VehicleCostPerUsage = vehicleCostPerUsage;
00180             OwnerID = ownerID;
00181             VehicleFlatCostForShortRouteLength = vehicleFlatCostForShortRouteLength;
00182             VehicleMaxRouteLengthForFlatCost = vehicleMaxRouteLengthForFlatCost;
00183         }
00184
00185         private static bool CanAccomodateCargoTypesTogether(IEnumerable<TransportRequest> requests)
00186         {
00187             foreach (var request in requests)
00188             {
00189                 if (requests.Any(req => request.CargoTypes.Any(ct =>
      req.RestrictedCargoTypes.Contains(ct))))
00190                 {
00191                     return false;
00192                 }
00193             }
00194             return true;
00195         }
00196
00197         private bool CanFitCapacity(IEnumerable<TransportRequest> requestGroup)
00198         {
00199             return Vehicle.CanFitCapacity(this.Capacity, this.CapacityAggregationType,
      requestGroup.Select(rq => rq.Size));
00200         }
00201
00202         internal double ComputeDistanceCost(double length)
00203         {
00204             if (length < this.VehicleMaxRouteLengthForFlatCost)
00205             {
00206                 return this.VehicleFlatCostForShortRouteLength;
00207             }
00208             else
00209             {
00210                 return this.VehicleCostPerDistanceUnit * length;
00211             }
00212         }
00213
00214         internal double ComputeTimeCost(double travelTime)
00215         {
00216             return this.VehicleCostPerTimeUnit * travelTime;
00217         }
00218
00226         public static bool CanFitCapacity(double[] capacity, Aggregation[] aggregationType,
```

```
    IEnumerable<double[]> sizes)
00227         {
00228             for (int i = 0; i < capacity.Length; i++)
00229             {
00230                 if (aggregationType[i] == Aggregation.Sum)
00231                 {
00232                     if (capacity[i] < sizes.Sum(r => r[i]))
00233                     {
00234                         return false;
00235                     }
00236                 }
00237                 else if (aggregationType[i] == Aggregation.Max)
00238                 {
00239                     if (capacity[i] < sizes.Max(r => r[i]))
00240                     {
00241                         return false;
00242                     }
00243                 }
00244                 else
00245                 {
00246                     throw new ArgumentException($"Vehicle does not support {aggregationType[i]}
    aggregation of sizes");
00247                 }
00248             }
00249             return true;
00250         }
00251
00257         public bool CanFitRequests(IEnumerable<TransportRequest> requests)
00258         {
00259             if (requests.Any(req => !this.CanHandleRequest(req)))
00260             {
00261                 return false;
00262             }
00263             if (!CanAccomodateCargoTypesTogether(requests))
00264             {
00265                 return false;
00266             }
00267             if (!CanFitCapacity(requests))
00268             {
00269                 return false;
00270             }
00271             return true;
00272         }
00273
00279         public bool CanFitRequestsSomewhereInVehicle(IEnumerable<TransportRequest> requests)
00280         {
00281             if (requests.Any(req => !this.CanHandleRequest(req)))
00282             {
00283                 return false;
00284             }
00285             foreach (var requestGroup in requests.GroupBy(rq => rq.PickupLocation.Id))
00286             {
00287                 if (!CanAccomodateCargoTypesTogether(requestGroup))
00288                 {
00289                     return false;
00290                 }
00291                 if (!CanFitCapacity(requestGroup))
00292                 {
00293                     return false;
00294                 }
00295             }
00296             foreach (var requestGroup in requests.GroupBy(rq => rq.DeliveryLocation.Id))
00297             {
00298                 if (!CanAccomodateCargoTypesTogether(requestGroup))
00299                 {
00300                     return false;
00301                 }
00302                 if (!CanFitCapacity(requestGroup))
00303                 {
00304                     return false;
00305                 }
00306             }
00307             return true;
00308         }
00309
00317         public bool CanHandleRequest(TransportRequest candidateRequest)
00318         {
00319             if (this.Capacity.Length != candidateRequest.Size.Length)
00320             {
00321                 throw new ArgumentException("Capacity properties of the vehicle do not match request
    size");
00322             }
00323             if (!candidateRequest.MaxVehicleSize.DoesVehicleFitIntoRestrictions(this.RoadProperties))
00324             {
00325                 return false;
00326             }
00327             for (int i = 0; i < this.Capacity.Length; i++)
```

```
00328             {
00329                 if (this.Capacity[i] < candidateRequest.Size[i])
00330                 {
00331                     return false;
00332                 }
00333             }
00334             for (int i = 0; i < candidateRequest.NecessaryVehicleSpecialProperties.Length; i++)
00335             {
00336                 if
      (!this.SpecialProperties.Contains(candidateRequest.NecessaryVehicleSpecialProperties[i]))
00337                 {
00338                     return false;
00339                 }
00340             }
00341             return true;
00342         }
00343     }
00344 }
```

7.57 VRPCostFunction.cs File Reference

Classes

- class VRPTWOptimizer.VRPCostFunction

  Class for calculating solution costs

Namespaces

- namespace VRPTWOptimizer

7.58 VRPCostFunction.cs

Go to the documentation of this file.

```
00001 using Newtonsoft.Json;
00002 using System;
00003 using System.Collections.Generic;
00004 using System.Linq;
00005 using VRPTWOptimizer.Interfaces;
00006
00007 namespace VRPTWOptimizer
00008 {
00012     public class VRPCostFunction
00013     {
00014         private const double DefaultDistanceFactor = 1e-6;
00015         private const double DefaultLateEarlyFactor = 1e-5;
00016         private const double DefaultLeftCargoFactor = 1e8;
00017         private const double DefaultRoutesCountFactor = 1e4;
00018         private static readonly Dictionary<int, double> DefaultCarrierFactor = new Dictionary<int,
      double>();
00022         public double CarrierMinDistanceFactor { get; set; }
00026         public Dictionary<int, double> CarrierMinDistanceThreshold { get; set; }
00030         public double CarrierShareFactor { get; set; }
00034         public Dictionary<int, double> CarrierShareRatio { get; set; }
00038         public double DistanceFactor { get; set; }
00042         public double DriveTimeFactor { get; set; }
00046         public double FillInFactor { get; set; }
00050         public double LeftCargoUnitFactor { get; set; }
00054         public double MaxDelayFactor { get; set; }
00058         public double MaxDelaySquaredFactor { get; set; }
00062         public double MaxEarlyArrivalFactor { get; set; }
00066         public double MaxEarlyArrivalSquaredFactor { get; set; }
00070         public double MaxVehicleSpreadFactor { get; set; }
00074         public double RoutesCountFactor { get; set; }
00078         public double TotalDelayFactor { get; set; }
00082         public double TotalDelaySquaredFactor { get; set; }
00086         public double TotalEarlyArrivalFactor { get; set; }
00090         public double TotalEarlyArrivalSquaredFactor { get; set; }
00094         public double UsageFactor { get; set; }
00095
```

```
00099         public VRPCostFunction()
00100         {
00101             CarrierMinDistanceThreshold = DefaultCarrierFactor;
00102             CarrierShareRatio = DefaultCarrierFactor;
00103             DistanceFactor = DefaultDistanceFactor;
00104             LeftCargoUnitFactor = DefaultLeftCargoFactor;
00105             MaxDelaySquaredFactor = DefaultLateEarlyFactor;
00106             MaxEarlyArrivalFactor = DefaultLateEarlyFactor;
00107             RoutesCountFactor = DefaultRoutesCountFactor;
00108         }
00109 
00132         [JsonConstructor]
00133         public VRPCostFunction(
00134             double distanceFactor,
00135             double usageFactor,
00136             double driveTimeFactor,
00137             double leftUnitFactor,
00138             double maxDelaySquaredFactor,
00139             double maxEarlyArrivalFactor,
00140             double totalDelaySquaredFactor,
00141             double totalEarlyArrivalFactor,
00142             double carrierMinDistanceFactor,
00143             double carrierShareFactor,
00144             double fillInFactor,
00145             Dictionary<int, double> carrierMinDistanceThreshold,
00146             Dictionary<int, double> carrierShareRatio,
00147             double routesCountFactor = 0,
00148             double totalEarlyArrivalSquaredFactor = 0,
00149             double totalDelayFactor = 0,
00150             double maxVehicleSpreadFactor = 0,
00151             double maxEarlyArrivalSquaredFactor = 0,
00152             double maxDelayFactor = 0)
00153         {
00154             DistanceFactor = distanceFactor;
00155             UsageFactor = usageFactor;
00156             DriveTimeFactor = driveTimeFactor;
00157             LeftCargoUnitFactor = leftUnitFactor;
00158             MaxDelaySquaredFactor = maxDelaySquaredFactor;
00159             MaxEarlyArrivalFactor = maxEarlyArrivalFactor;
00160             TotalDelaySquaredFactor = totalDelaySquaredFactor;
00161             TotalEarlyArrivalFactor = totalEarlyArrivalFactor;
00162             CarrierMinDistanceFactor = carrierMinDistanceFactor;
00163             CarrierMinDistanceThreshold = carrierMinDistanceThreshold;
00164             CarrierShareFactor = carrierShareFactor;
00165             CarrierShareRatio = carrierShareRatio;
00166             FillInFactor = fillInFactor;
00167             RoutesCountFactor = routesCountFactor;
00168             TotalEarlyArrivalSquaredFactor = totalEarlyArrivalSquaredFactor;
00169             TotalDelayFactor = totalDelayFactor;
00170             MaxVehicleSpreadFactor = maxVehicleSpreadFactor;
00171             MaxEarlyArrivalSquaredFactor = maxEarlyArrivalSquaredFactor;
00172             MaxDelayFactor = maxDelayFactor;
00173         }
00174 
00181         public static double ComputeFillInFactor(List<double[]> cargoOnRouteStart, double[]
      vehicleCapacity, Enums.Aggregation[] aggregationType)
00182         {
00183             double maxFillIn = 0.0;
00184             for (int i = 0; i < vehicleCapacity.Length; i++)
00185             {
00186                 double cargoSumI = 0.0;
00187                 if (aggregationType[i] == Enums.Aggregation.Sum)
00188                 {
00189                     cargoSumI = cargoOnRouteStart.Sum(c => c[i]);
00190                 }
00191                 else if (aggregationType[i] == Enums.Aggregation.Max)
00192                 {
00193                     if (cargoOnRouteStart.Any())
00194                     {
00195                         cargoSumI = cargoOnRouteStart.Max(c => c[i]);
00196                     }
00197                 }
00198                 maxFillIn = Math.Max(maxFillIn, cargoSumI / vehicleCapacity[i]);
00199             }
00200             return maxFillIn;
00201         }
00202 
00208         public static double ComputeFillInFactor(IRoute route)
00209         {
00210             List<double[]> cargoOnRouteStart = route.LoadedRequests[0].Select(rq => rq.Size).ToList();
00211             double[] vehicleCapacity = route.Vehicle.Capacity;
00212             var aggregationType = route.Vehicle.CapacityAggregationType;
00213             return ComputeFillInFactor(cargoOnRouteStart, vehicleCapacity, aggregationType);
00214         }
00215 
00221         public static double ComputeMaxEarlyArrival(IRoute route)
00222         {
```

```
00223             double[] timeWindowsStart = route.TimeWindowStart
00224                 .Select(tws => tws).ToArray();
00225             double[] arrivalTimes = route.ArrivalTimes
00226                 .Select(art => art).ToArray();
00227             return ComputeMaxTimeDiff(timeWindowsStart, arrivalTimes, true);
00228         }
00229 
00237         public static double ComputeMaxTimeDiff(double[] referenceValues, double[] trueValues, bool
      referenceIsLowerBound)
00238         {
00239             double timeDiff = 0.0;
00240             for (int i = 0; i < referenceValues.Length; i++)
00241             {
00242                 if (referenceIsLowerBound)
00243                 {
00244                     timeDiff = Math.Max(timeDiff, referenceValues[i] - trueValues[i]);
00245                 }
00246                 else
00247                 {
00248                     timeDiff = Math.Max(timeDiff, trueValues[i] - referenceValues[i]);
00249                 }
00250             }
00251             return timeDiff;
00252         }
00253 
00259         public static double ComputeTotalEarlyArrival(IRoute route)
00260         {
00261             double[] timeWindowsStart = route.TimeWindowStart
00262                 .Select(tws => tws).ToArray();
00263             double[] arrivalTimes = route.ArrivalTimes
00264                 .Select(art => art).ToArray();
00265             return ComputeTotalTimeDiff(timeWindowsStart, arrivalTimes, true);
00266         }
00267 
00275         public static double ComputeTotalTimeDiff(double[] referenceValues, double[] trueValues, bool
      referenceIsLowerBound)
00276         {
00277             double timeDiff = 0.0;
00278             for (int i = 0; i < referenceValues.Length; i++)
00279             {
00280                 if (referenceIsLowerBound)
00281                 {
00282                     timeDiff += Math.Max(0, referenceValues[i] - trueValues[i]);
00283                 }
00284                 else
00285                 {
00286                     timeDiff += Math.Max(0, trueValues[i] - referenceValues[i]);
00287                 }
00288             }
00289             return timeDiff;
00290         }
00291 
00299         public static VRPCostFunction GetDefaultParametersFunction()
00300         {
00301             return new(
00302                 carrierMinDistanceFactor:  0.0,
00303                 carrierMinDistanceThreshold:  DefaultCarrierFactor,
00304                 carrierShareFactor:  0.0,
00305                 carrierShareRatio:  DefaultCarrierFactor,
00306                 distanceFactor:  DefaultDistanceFactor,
00307                 driveTimeFactor:  0.0,
00308                 fillInFactor:  0.0,
00309                 leftUnitFactor:  DefaultLeftCargoFactor,
00310                 maxDelaySquaredFactor:  DefaultLateEarlyFactor,
00311                 maxEarlyArrivalFactor:  DefaultLateEarlyFactor,
00312                 totalDelaySquaredFactor:  0.0,
00313                 totalEarlyArrivalFactor:  0.0,
00314                 routesCountFactor:  DefaultRoutesCountFactor,
00315                 usageFactor:  0.0
00316                 );
00317         }
00318 
00324         public double SingleRouteValue(IRoute route)
00325         {
00326             double length = route.Length;
00327             double travelTime = route.TravelTime;
00328             Vehicle vehicle = route.Vehicle;
00329             Vehicle vehicleTractor = route.VehicleTractor;
00330             IEnumerable<double> revenueValues = route.LoadedRequests.SelectMany(rq => rq.Select(rq =>
      rq.RevenueValue));
00331             double[] timeWindowsStart = route.TimeWindowStart
00332                 .Select(tws => tws).ToArray();
00333             double[] arrivalTimes = route.ArrivalTimes
00334                 .Select(art => art).ToArray();
00335             double vehicleDistanceCost = vehicle.ComputeDistanceCost(length);
00336             double vehicleTimeCost = vehicle.ComputeTimeCost(travelTime);
00337             if (vehicleTractor != null)
```

```
00338             {
00339                 vehicleDistanceCost += route.VehicleTractor.ComputeDistanceCost(length);
00340                 vehicleTimeCost += route.VehicleTractor.ComputeTimeCost(travelTime);
00341             }
00342             //cost of being too early
00343             double routeTotalEarly = ComputeTotalTimeDiff(timeWindowsStart, arrivalTimes, true);
00344             double earlyArrivalsCost = TotalEarlyArrivalFactor * routeTotalEarly +
      TotalEarlyArrivalSquaredFactor * routeTotalEarly * routeTotalEarly;
00345             //cost of delay
00346             double routeTotalDelay = route.TotalDelay;
00347             double lateArrivalsCost = TotalDelayFactor * routeTotalDelay + TotalDelaySquaredFactor *
      routeTotalDelay * routeTotalDelay;
00348             return
00349                 DistanceFactor * vehicleDistanceCost +
00350                 //cost of cargo cooling
00351                 DriveTimeFactor * vehicleTimeCost +
00352                 //cost of using vehicle at all
00353                 lateArrivalsCost +
00354                 earlyArrivalsCost -
00355                 //gain of serving profit generating requests
00356                 revenueValues.Sum(val => val);
00357             ;
00358         }
00359
00366         public double Value(List<IRoute> routes, List<TransportRequest> leftRequests)
00367         {
00368             /*
00369 *              double vehicleUsageCost = route.Vehicle.VehicleCostPerUsage;
00370 vehicleUsageCost += route.VehicleTractor.VehicleCostPerUsage;
00371
00372 */
00373             var orderedRoutes = routes.OrderBy(rt => rt.VehicleTractor != null ?  rt.VehicleTractor.Id
      :  rt.Vehicle.Id)
00374                 .ThenBy(rt => rt.ArrivalTimes[0]);
00375             IRoute previousRoute = null;
00376             double maxVehicleSpread = 0.0;
00377             foreach (var currentRoute in orderedRoutes)
00378             {
00379                 if (previousRoute != null)
00380                 {
00381                     if (currentRoute.VehicleTractor != null)
00382                     {
00383                         if (previousRoute.VehicleTractor != null && currentRoute.VehicleTractor.Id ==
      previousRoute.VehicleTractor.Id)
00384                         {
00385                             maxVehicleSpread = Math.Max(maxVehicleSpread,
      currentRoute.DepartureTimes[0] - previousRoute.ArrivalTimes[^1]);
00386                         }
00387                     }
00388                     else if (currentRoute.Vehicle.Id == previousRoute.Vehicle.Id)
00389                     {
00390                         maxVehicleSpread = Math.Max(maxVehicleSpread, currentRoute.ArrivalTimes[0] -
      previousRoute.DepartureTimes[^1]);
00391                     }
00392                 }
00393                 previousRoute = currentRoute;
00394             }
00395             double maxVehicleSpreadCost = MaxVehicleSpreadFactor * maxVehicleSpread;
00396
00397             var realDistanceDict = routes
00398                 .GroupBy(rt => rt.Vehicle.OwnerID)
00399                 .ToDictionary(gr => gr.Key, gr => gr.Sum(rt => rt.Length));
00400             var totalDistance = routes.Sum(rt => rt.Length);
00401             var lowFillInCost = routes.Any() ?  FillInFactor * routes.Average(route => (1 -
      ComputeFillInFactor(route))) :  0.0;
00402             var maxEarlyArrivalCost = routes.Any() ?  routes.Max(route =>
00403             {
00404                 double routeMaxEarlyArrival = ComputeMaxEarlyArrival(route);
00405                 return MaxEarlyArrivalFactor * routeMaxEarlyArrival + MaxEarlyArrivalSquaredFactor *
      routeMaxEarlyArrival * routeMaxEarlyArrival;
00406             }) :  0.0;
00407             var maxLaterArrivalCost = routes.Any() ?  routes.Max(route =>
00408             {
00409                 double maxDelay = route.MaxDelay;
00410                 return MaxDelayFactor * maxDelay + MaxDelaySquaredFactor * maxDelay * maxDelay;
00411             }) :  0.0;
00412             var routesCost = routes.Sum(
00413                             //cost of routes
00414                             rt => SingleRouteValue(rt));
00415             //cost of not delivering certain amount of cargo
00416             var leftCargoCost = LeftCargoUnitFactor * leftRequests.Sum(lr => lr.PackageCount);
00417             var carrierCost =  //cost of not getting equal routes distance share between car owners
00418                 routes.Any() ?  CarrierShareFactor * CarrierShareRatio.Keys.Sum(csrDictKey =>
00419                     Math.Abs(totalDistance * CarrierShareRatio[csrDictKey] -
      realDistanceDict[csrDictKey]) / 2.0) +
00420                 //cost of not providing enough kilometers per car owner
00421                 CarrierMinDistanceFactor * CarrierMinDistanceThreshold.Keys.Sum(cmdtDictKey =>
```

```
00422                     Math.Abs(Math.Max(0, CarrierMinDistanceThreshold[cmdtDictKey] -
      realDistanceDict[cmdtDictKey]))) :  0.0;
00423             double routeCountCost = RoutesCountFactor * routes.Count;
00424             return routesCost +
00425                 lowFillInCost +
00426                 routeCountCost +
00427                 maxVehicleSpreadCost +
00428                 maxEarlyArrivalCost +
00429                 maxLaterArrivalCost +
00430                 carrierCost +
00431                 leftCargoCost;
00432         }
00433     }
00434 }
```

7.59 VRPDefinition.cs File Reference

Classes

- class VRPTWOptimizer.VRPDefinition

  Describes data for a generalized Vehicle Routing Problem

Namespaces

- namespace VRPTWOptimizer

7.60 VRPDefinition.cs

Go to the documentation of this file.

```
00001 using CommonGIS.Interfaces;
00002 using Newtonsoft.Json;
00003 using System;
00004 using System.Collections.Generic;
00005 using System.IO;
00006 using System.Reflection;
00007 using VRPTWOptimizer.Interfaces;
00008
00009 namespace VRPTWOptimizer
00010 {
00014     public class VRPDefinition
00015     {
00016         private class LawAbidingFloatConverter :  JsonConverter
00017         {
00018             public override bool CanRead
00019             {
00020                 get
00021                 {
00022                     return false;
00023                 }
00024             }
00025             public override bool CanWrite
00026             {
00027                 get
00028                 {
00029                     return true;
00030                 }
00031             }
00032
00033             public override bool CanConvert(Type objectType)
00034             {
00035                 return objectType == typeof(double) || objectType == typeof(float);
00036             }
00037
00038             public override object ReadJson(JsonReader reader, Type objectType, object existingValue,
      JsonSerializer serializer)
00039             {
00040                 throw new NotImplementedException();
00041             }
00042
```

```
00043             public override void WriteJson(JsonWriter writer, object value, JsonSerializer serializer)
00044             {
00045                 var val = value as double?  ??  (double?)(value as float?);
00046                 if (val == null || Double.IsNaN((double)val) || Double.IsInfinity((double)val))
00047                 {
00048                     writer.WriteNull();
00049                     return;
00050                 }
00051                 writer.WriteValue((double)val);
00052             }
00053         }
00054 
00058         public string Client { get; set; }
00062         public VRPCostFunction CostFunctionFactors { get; set; }
00066         public Version DataFormatVersion => Assembly.GetAssembly(new
      VRPDefinition().GetType()).GetName().Version;
00067 
00071         public DateTime Date { get; set; }
00075         public string DepotId { get; set; }
00079         public IDistanceProvider DistanceData { get; set; }
00083         public List<Driver> Drivers { get; set; }
00087         public List<TransportRequest> Requests { get; set; }
00091         public ITimeEstimator ServiceTimeEstimator { get; set; }
00095         public List<VRPSolution> Solutions { get; set; }
00099         public List<Vehicle> Vehicles { get; set; }
00103         public DateTime ZeroHour { get; set; }
00104 
00114         [Obsolete("Please use GenerateVRPDefintion with conscious costFunctionFactors definition")]
00115         public static VRPDefinition GenerateVRPDefintion(
00116             IVRPProvider vrpProvider,
00117             DateTime billingDate,
00118             IDistanceProvider distanceProvider,
00119             ITimeEstimator timeEstimator,
00120             string client)
00121         {
00122             VRPCostFunction costFunctionFactors = VRPCostFunction.GetDefaultParametersFunction();
00123             return GenerateVRPDefintion(
00124                 vrpProvider,
00125                 costFunctionFactors,
00126                 billingDate,
00127                 distanceProvider,
00128                 timeEstimator,
00129                 client);
00130         }
00131 
00142         public static VRPDefinition GenerateVRPDefintion(
00143             IVRPProvider vrpProvider,
00144             VRPCostFunction costFunctionFactors,
00145             DateTime billingDate,
00146             IDistanceProvider distanceProvider,
00147             ITimeEstimator timeEstimator,
00148             string client)
00149         {
00150             List<VRPSolution> vrpSolutions = new();
00151             VRPDefinition vrpDefinition = new()
00152             {
00153                 CostFunctionFactors = costFunctionFactors,
00154                 Requests = vrpProvider.Requests,
00155                 ServiceTimeEstimator = timeEstimator,
00156                 Vehicles = vrpProvider.Vehicles,
00157                 Drivers = vrpProvider.Drivers,
00158                 Solutions = vrpSolutions,
00159                 Date = billingDate.Date,
00160                 DepotId = vrpProvider.HomeDepot.Id,
00161                 Client = client,
00162                 ZeroHour = vrpProvider.ZeroHour,
00163                 DistanceData = distanceProvider
00164             };
00165 
00166             return vrpDefinition;
00167         }
00168 
00173         public void AddSolution(VRPSolution vrpSolution)
00174         {
00175             if (Solutions == null)
00176             {
00177                 Solutions = new();
00178             }
00179             Solutions.Add(vrpSolution);
00180         }
00181 
00186         public string ToPrettyJSONString()
00187         {
00188             var settings = new JsonSerializerSettings();
00189             var floatConverter = new LawAbidingFloatConverter();
00190             settings.Converters.Add(floatConverter);
00191             settings.Formatting = Formatting.Indented;
```

```
00192             var serializer = JsonSerializer.Create(settings);
00193             var writer = new StringWriter();
00194             serializer.Serialize(writer, this);
00195             return writer.ToString();
00196         }
00197
00203         public bool TrySaveToFile(string filename)
00204         {
00205             string contents = this.ToPrettyJSONString();
00206             try
00207             {
00208                 File.WriteAllText(filename, contents);
00209                 return true;
00210             }
00211             catch (UnauthorizedAccessException)
00212             {
00213                 Console.WriteLine($"Cannot access {filename}");
00214                 Console.Write(contents);
00215                 return false;
00216             }
00217             catch (DirectoryNotFoundException)
00218             {
00219                 Console.WriteLine($"Designated location directory {filename} does not exists");
00220                 Console.Write(contents);
00221                 return false;
00222             }
00223             catch (IOException)
00224             {
00225                 Console.WriteLine($"Cannot create {filename} for unknown reason");
00226                 Console.Write(contents);
00227                 return false;
00228             }
00229         }
00230     }
00231 }
```

7.61 VRPOptimizerResult.cs File Reference

Classes

- class VRPTWOptimizer.VRPOptimizerResult

  Represents the output of Vehicle Routing Problem optimization algorithm

Namespaces

- namespace VRPTWOptimizer

7.62 VRPOptimizerResult.cs

Go to the documentation of this file.

```
00001 using System.Collections.Generic;
00002 using VRPTWOptimizer.Interfaces;
00003
00004 namespace VRPTWOptimizer
00005 {
00009     public class VRPOptimizerResult
00010     {
00014         public List<TransportRequest> LeftRequests { get; set; }
00015
00019         public List<IRoute> Routes { get; set; }
00020     }
00021 }
```

7.63 VRPSolution.cs File Reference

Classes

- class VRPTWOptimizer.VRPSolution

  Definition of structure describing solution to the Vehicle Routing Problem. Includes Vehicle assignment to TransportRequest and Vehicle schedule
- class VRPTWOptimizer.VRPSolution.ScheduleItem

  Single time entry describing planned visit at a given Location
- class VRPTWOptimizer.VRPSolution.TransportItem

  Entry describing a single loop of the Vehicle/combined Vehicle

Namespaces

- namespace VRPTWOptimizer

7.64 VRPSolution.cs

Go to the documentation of this file.

```
00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using System.Reflection;
00005 using VRPTWOptimizer.Interfaces;
00006
00007 namespace VRPTWOptimizer
00008 {
00013     public class VRPSolution
00014     {
00018         public class ScheduleItem
00019         {
00023             public double ArrivalTime { get; set; }
00027             public double Delay { get; set; }
00031             public double DepartureTime { get; set; }
00035             public List<int> LoadedRequestsIds { get; set; }
00039             public string LocationId { get; set; }
00040
00044             public List<int> UnloadedRequestsIds { get; set; }
00045         }
00046
00050         public class TransportItem
00051         {
00055             public double AvailableForLoadingTime { get; set; }
00059             public double AvailableForNextAssignmentTime { get; set; }
00064             public int DriverId { get; set; }
00068             public double FillInRatio { get; set; }
00072             public double Length { get; set; }
00076             public List<ScheduleItem> Schedule { get; set; }
00080             public int TractorId { get; set; }
00084             public int TrailerTruckId { get; set; }
00088             public int TransportId { get; set; }
00089         }
00090
00094         public string Algorithm { get; set; }
00098         public double ComputationTime { get; set; }
00102         public DateTime ComputationTimestamp { get; set; }
00106         public string ComputerId { get; set; }
00110         public int DelaysCount { get; set; }
00114         public List<int> LeftRequestsIds { get; set; }
00118         public double MaxDelay { get; set; }
00122         public double TotalDelay { get; set; }
00126         public double TotalLength { get; set; }
00130         public List<TransportItem> Transports { get; set; }
00134         public Version Version { get; set; }
00135
00145         public static VRPSolution GenerateVRPSolution(
00146             IVRPOptimizer _optimizer,
00147             DateTime computationsStart,
00148             DateTime computationsEnd,
```

```
00149             List<TransportRequest> leftRequests,
00150             List<IRoute> routes)
00151         {
00152             List<VRPSolution.TransportItem> transportItems = new();
00153             var orderedTrailerAssignment = routes
00154                 .OrderBy(rt => rt.ArrivalTimes[0]);
00155             int transportId = 1;
00156             foreach (var assignment in orderedTrailerAssignment)
00157             {
00158                 List<VRPSolution.ScheduleItem> scheduleItems = new();
00159                 for (int i = 0; i < assignment.VisitedLocations.Count; i++)
00160                 {
00161                     VRPSolution.ScheduleItem scheduleItem = new()
00162                     {
00163                         LocationId = assignment.VisitedLocations[i].Id,
00164                         ArrivalTime = assignment.ArrivalTimes[i],
00165                         DepartureTime = assignment.DepartureTimes[i],
00166                         Delay = Math.Max(assignment.ArrivalTimes[i] - assignment.TimeWindowEnd[i], 0),
00167                         LoadedRequestsIds = assignment.LoadedRequests[i].Select(rq => rq.Id).ToList(),
00168                         UnloadedRequestsIds = assignment.UnloadedRequests[i].Select(rq =>
      rq.Id).ToList()
00169                     };
00170                     scheduleItems.Add(scheduleItem);
00171                 }
00172                 double fillInRatio = assignment.Vehicle.Capacity
00173                     .Select((capacity, index) => index)
00174                     .Max(index => assignment.LoadedRequests[0].Sum(rq => rq.Size[index]) /
      assignment.Vehicle.Capacity[index])
00175                     ;
00176                 VRPSolution.TransportItem transport = new()
00177                 {
00178                     TransportId = transportId,
00179                     TractorId = assignment.VehicleTractor == null ?  -1 :
      assignment.VehicleTractor.Id,
00180                     TrailerTruckId = assignment.Vehicle.Id,
00181                     DriverId = assignment.VehicleDriver == null ?  -1 :  assignment.VehicleDriver.Id,
00182                     Length = assignment.Length,
00183                     Schedule = scheduleItems,
00184                     AvailableForLoadingTime = assignment.ArrivalTimes[0],
00185                     AvailableForNextAssignmentTime = assignment.DepartureTimes[^1],
00186                     FillInRatio = Math.Round(fillInRatio, 2)
00187                 };
00188                 transportItems.Add(transport);
00189                 transportId++;
00190             }
00191
00192             VRPSolution vrpSolution = new()
00193             {
00194                 LeftRequestsIds = leftRequests.Select(req => req.Id).ToList(),
00195                 ComputationTimestamp = computationsEnd,
00196                 ComputationTime = (computationsEnd - computationsStart).TotalSeconds,
00197                 ComputerId = Environment.MachineName,
00198                 Version = Assembly.GetAssembly(_optimizer.GetType()).GetName().Version,
00199                 DelaysCount = routes.Count(ta => ta.TotalDelay >= 0),
00200                 MaxDelay = routes.Any() ?  routes.Max(ta => ta.MaxDelay) :  0,
00201                 TotalDelay = routes.Sum(ta => ta.TotalDelay),
00202                 TotalLength = routes.Sum(ta => ta.Length),
00203                 Transports = transportItems,
00204                 Algorithm = _optimizer.GetType().FullName
00205             };
00206             return vrpSolution;
00207         }
00208     }
00209 }
```

Index