FleetEngine 1.14

A Vehicle Routing Planner SDK

Table of Contents

Part I

Welcome

1 Welcome

FleetEngine 1.14

A server for solving generalized vehicle routing problems (VRP).

It has an open API which allows you to integrate it with almost any software system.

Main optimizing features

- Input / output in XML format with XSD schemas available
- Multiple depots
- Multiple capacities
- Multiple resource types (non-homogenous fleet)
- Resource specific properties
- Service time (flexible definition)
- Time windows
- Job priorities
- Standby jobs
- Breaks
- Load balancing
- Incompatible jobs (sheep-lion type of restriction)
- Mixed pickup and delivery
- Job time intervals

Built-in time & distance matrix calculator

- Asymmetric matrix
- One-way streets
- Turn restrictions
- Dynamic segmentation
- Limit routes to vehicles < x tons etc.
- Driving directions

1.1 Overview

FleetEngine is an application for optimizing a fleet of vehicles (resources) with a list of jobs, so the total cost is minimized by using as few vehicles as possible, driving as few km as possible etc, while observing a number of restrictions, such as capacity, time windows etc.

The main principle in the optimization process (a "Task") is having a list of jobs and a list of resources & resource periods. These are described through XML. Optimisation is usually based on distances travelled along a street network, but can also use straight line distances.

FleetEngine outputs XML containing an optimised sequence of jobs and data to allow its visualisation.

The main elements in a task are:

Resource types
These define each type of available resource and attributes of that resource such as cost, competences, capabilities etc.

Resources
These are 'instances' of resource types to be used in the analysis. This can typically be a specific person or vehicle.

Resource periods
These define availability of resources. A resource period is the period of time that a resource is available to service requesting jobs. This also details the resource 'start' location and required 'stop' location at the end of the resource period (e.g. depot locations such as home or work).

Depots
These are possible start and end locations for resources. These may typically be home or work locations (or any given location). For example, where a lorry driver starts and ends the working day.

Jobs
The actual jobs that must be serviced. Jobs can have particular 'demand' requirements and can only be serviced by resources that have sufficient capacity / competence. Service time is the time taken to complete the job at the location. There can be a service time for both the pickup and delivery part of the job.

Further details can be seen in the chapter about input XML 25.

1.2 Applications overview

FleetEngine consists of 6 applications:

Network Import 15
This is for setting up a street network (SHP / MIF / TAB) for use in the optimizer. A network is optional.

FleetEngine SOAP server 15
The main optimizing application.

Monitor 18
This allows you to monitor status of all tasks and start / stop individual tasks on the SOAP server.

FleetEngine Desktop optimizer 20
This allows you to run single tasks and easily follow progress. Good for ad-hoc tasks and testing. Allows calling the viewer application as an integrated part.

FleetEngine COM library 22
This is a slightly simpler COM version of the SOAP server, allowing optimizing a single task at a time.

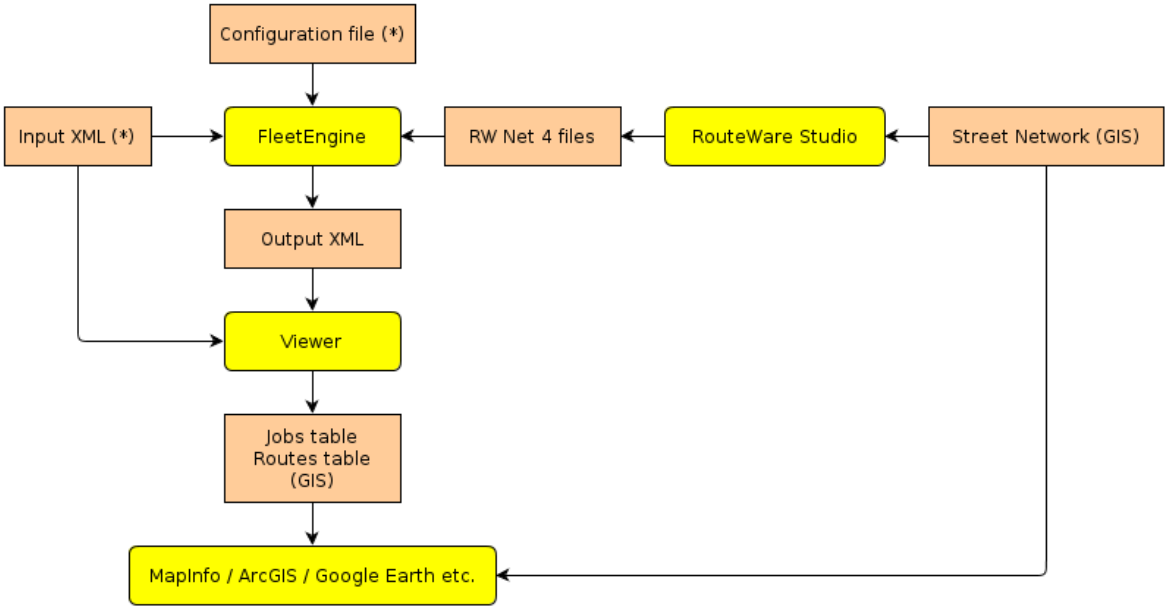
Viewer 24
This allows to you to merge the input and output from a calculation and generate GIS-files that can be viewed in your favourite GIS.

SOAP server, Desktop optimizer and COM library are all fully self-contained and do not require any of the other applications.

SOAP server and Desktop optimizer are available as 64-bit applications too.

1.3 Data flow

This chart shows how data flows between the various applications in FleetEngine.



(*) Always required.

GIS means a SHP / TAB / MIF table. As output from Viewer it can also be KML / GML.

1.4 Optimization

In OR-terms we solve the generalized vehicle routing problem (VRP). Not to optimality, but rather trying to find a "good" solution.

4 main kind of tasks are supported:

1. On-site - nothing is moved around
2. Pickup at start of route and and delivery at different locations
3. Pickup at different locations and delivery at the end of the route
4. Mixed pickup and delivery, where items can be picked up and delivered along the route

There is a lot of flexibility added:

- A resource period can start / stop at different depots (office, home address etc) or being open ended
- Resource types (typically vehicles) can have different capacities and costs
- Resources and jobs can have competences associated with them, so only resources with the correct competence gets matched with a job
- A job can pick up goods from one location and deliver it to another location (mixed pickup and delivery)
- A job can have multiple time windows defined (execute the job during one time window or another)
- Service time (time to execute a job) can be defined very flexible

This setup covers a lot of scheduling tasks in many industries, such as school bus routing, scheduling of maintenance jobs, delivery of various goods etc. In the chapter describing the input

XML 25 these are explained in more detail.

Target
To minimize cost as defined by the resource types.

1.5 What can't be done !!

Besides the list of features shown on the first page and explained in the chapter about input XML 25, it is sometimes worth having explained what isn't possible. It may help getting a better understanding of FleetEngine capabilities:

Generally

- An end user interface is not included. Users are supposed to build this on their own.

Features not (yet) supported, but which will be added later on:

- "Soft" restrictions as opposed to "hard" restrictions.
- Multi-core support for faster calculations.
- Different drive times throughout the day (peak / off-peak hours).

These features fall outside the framework

- A single job can not be split between multiple resource periods or between resources.
- A single job can not be split across multiple time windows.

1.6 Getting started

This section contains a step-by-step instruction for getting started as easily as possible with the sample tasks.

SOAP server 15

- Run Fleetengine.exe in the Binary folder. Click the "Start" button.
- Run FE_CSharp_Test.exe in the C# folder.
- Navigate to the input file ..\Sample Problems\c101\c101.xml.
- Set the output file name.
- Click Start button.

Desktop optimizer 20

- Run FleetEngineDesktop.exe in the Binary folder.
- Navigate to the input file ..\Sample Problems\c101\c101.xml.
- Edit the output file name if needed.
- Click OK button.

COM library 22

- Register the COM library by running the register_com.bat file in the Binary folder.
- Run FE_COM_Test.exe in the VB 6 client folder.
- Edit input / output file names if needed.

- Click Start button.

- Now watch as the optimization improves the solution. Very soon (after ~10 sec) a cost of 100828 has been reached. This is the same as the optimum solution.
- Click the Stop / Abort button
- Start fexml2gis.exe in the binary folder. Locate the input file. Output file is automatically setup, edit if needed. Choose MIF / SHP / TAB as output format.
- Click the "Write to GIS" button.
- If you choose SHP and have ArcGIS 10 installed, you can open the mxd document and see the result.
- If you choose TAB and have MapInfo installed, you can open the workspace and see the result.

You can do a similar experiment with this file: "..\sample problems\tiger\sample.xml". Just change the input/output file names.

1.7 System requirements

Operating System
FleetEngine runs on most recent flavors of Windows. It requires .NET 4 and MS XML 6 to be installed.

CPU
Most operations are CPU-bound, so a fast CPU is a good idea, but there are no specific requirements.
If you want to properly handle multiple tasks at a time, one core for each task is recommended.

RAM
Networks require appr. 50 bytes per link. An example: A network with 1 million link occupies 50 MB.

The optimization part generally requires little memory, but if you have many jobs the distance and time matrices can get very big.

Requirement: (64 x "Resource types" x (Depots + 2 x Jobs) ^ 2) bytes

Example: 2 resource types, 24 depots and 1000 jobs: 500 MB

We have included a spreadsheet, which makes it easy to calculate the required memory.

Additional software
An XML parser for the output is required.

An application like XMLspy is a good idea for setting up data, validating against the schemas etc.

A GIS application is required to easily view the output from the supplied Viewer 24 (a .NET 4 application).

1.8 Versions

This table describes the various versions available:

Version	Usage	Cores	Installations	Run-time
Full	In-house	2	1	Yes
Full, site	In-house	All	Multiple	Yes

Full SP	Service provider	All	1	No
Full SP, site	Service provider	All	Multiple	No
RW Net 4 Pro		1	1	No

RW Net 4 version 37
This is a single ResourcePeriod version, supplied for free with RW Net 4 Pro.

In-house versions
The in-house versions only allow you to run optimizations for a single fleet (typically your own). This version includes the licensee's name as part of the XML output.

SP version
This allow you to run optimizations for multiple fleets.
It also allows you to provide access to the server (over the internet) for other customers.

Cores
This refers to the number of CPU-cores the software will use. Unless you calculate multiple jobs at one time and have >1 core in your computer, this has no influence currently. This is planned to change in the future, when support for parallelization is added.

Installations
This refers to the number of installations of the software allowed at your site. This includes running multiple copies of the optimizer at one time. I.e. you are only allowed to start one instance of the application (SOAP Server, COM or Desktop), unless you have a site license.

Run-time license
This is a license, where we only provide support to the original license holder. Requires one full license and additional run-time licenses are to be bundled as part of a larger software package. Price is 30% of full price.

Updates
Minor updates are provided free for all.
Major updates are only free, if you are on the 1 or 2 year license models.

1.9 Tips

We have collected some tips / general advice:

General advice

- How long time does a calculation take? Key factors in increased processing time are total number of jobs and especially number of jobs assigned to each resource period. You should try to keep number of jobs per resource period below 100. Total number of jobs can not exceed 7500 and depending upon other settings, the limit may be lower. For tasks that big you may be looking at run times of couple of days. Consider aggregating jobs in advance if possible or split the task into smaller geographical areas. Other factors such as geographical spread, time windows, "tightness" of jobs etc. also play a major role.

- When should a calculation be stopped? From an optimization point of view when there has been "several" iterations without improvements. Use the StopRatio property in the XML to aid in defining this.

- As we make changes to FleetEngine (improving it / adding features etc) it may also happen that future results for the same input gets worse. This is inevitable in a system, which relies on randomized heuristics.

- Deciding if FleetEngine can handle a specific optimization task can sometimes be tricky, since the requirements to a solution may not be known in detail. Don't underestimate the resources to investigate this part !

More technical advice

- Use as few job time windows as possible. If the only resource period is from 8:00 to 11:00, there is no need to specify that the job should be executed between 7:00 and 12:00. It only makes the calculations slower. Generally use as few restrictions as possible to describe the problem.
- We recommend starting with not so many jobs (if you have many), since that will make it faster to detect if a task need to be formulated differently or a restriction is missing.
- Cost may increase (!) during an optimization process, if previously unassigned jobs get assigned. This is not an error.

1.10 License terms

License terms for FleetEngine, Apr 2011

All copyrights belong to RouteWare.

Disassemble or reverse engineering of FleetEngine binaries is not allowed.

Licensor is allowed to make copies of FleetEngine for backup purposes only.

Licensor is not allowed to sell or in any other way hand over the right to use the software to any other party.

RouteWare is not responsible for any problems, direct or indirect, which FleetEngine may cause - no matter what the reason may be.

Any problem / error will be corrected as fast as possible within normal business hours. If RouteWare is not able to correct problems, which to a severe degree affect the functionality of the software, a refund is made, which matches the degree to which the software doesn't function properly. This refund is based on what the licensor has paid within the previous 12 months and cannot exceed this amount.

Licensor gets free updates for all minor updates (1.x versions). If licensor has a time-limited license (typically 1 or 2 years), there is also free access to major updates within the licensing period.

Licensor is allowed to run as many instances of FleetEngine as the license allows for production use. This is either 1 (non-site licenses) or multiple (site licenses).

Licensor may also install it on 1 or more development computers, which is only being used for testing / development.

1.11 History

This list is only about new features. See history.txt document for details on bug fixes.

Version 1.14 (8. Jan 2015)
Added UTurnTime to prevent too many u-turns
Added StopRatio as new stopping mechanism
Additional output information
Optional saving of graph in desktop version to PNG.

Version 1.13 (11. Sep 2013)
Server: Task garbage collection
Server: Distance matrix progress event
Light version has been removed.

Version 1.12 (14. Aug 2013)
Preferred resources

Version 1.11 (19. May 2013)
Server split into .NET executable and native DLL. WCF compatible.
This requires a new client, since wsdl has changed.
Uses MS XML 6 instead of MS XML 4
64-bit version of server and desktop version

Version 1.10 (21. Mar 2013)
Job time intervals (visit customer every N days)

Version 1.09 (25. Oct 2012)
Alternative time windows for the same job can have different costs.
Network Importer application replaced by RouteWare Studio.

Version 1.08 (25. Sep 2012)
No new features

Version 1.07 (4. Jun 2012)
Initial solution is now much better, meaning you get good quality solutions in shorter time.

Version 1.06 (20. Mar 2012)
Desktop version has built-in GIS viewer functionality
Delivery tasks can be formulated without using pickup node as proxy
Various performance optimizations

Version 1.05 (15. Jan 2012)
Support in optimization for incompatible jobs
Driving directions in XML output
Option to use warning on infeasible jobs
Viewer library expanded
RW Net 4 license option

Version 1.04 (28. Nov 2011)
No new features

Version 1.03 (9. Nov 2011)
Support in optimization for job priorities, standby jobs, breaks and work load balancing
GIS output options in XML files
New GIS viewer
ArcGIS sample added
Updated documentation in many places
Schemas has been updated to be more strict on sign of numbers
GUI version renamed as Desktop version

Version 1.02 (18. Aug 2011)
Performance improvements (up to a factor 3 for some tasks)
Support for no-drive-through bits in attribute field
Network encryption

Version 1.01 (10. May 2011)
Added new options for job and directions - pickup and delivery

Version 1.00 (11. Apr 2011)

Part II

User Manual

2 User Manual

2.1 Network Import

You can skip this step, if you either provide the time / distance matrix as part of the input XML or use as-the-crow-fly distances.

Download and install RouteWare Studio from our website in order to import TAB / SHP / MIF files into RW Net 4 format, which is used in FleetEngine.

You will have to use your fleetengine.lic file in RouteWare Studio.

2.2 SOAP server

The server works by sending it a single XML file as task description from a client. After that queries can be sent to ask status of the task. Once completed, you can extract the result as a new XML file, get the current result or simply stop the calculations and get the result.

The server stops when MaxCalculationTime, as specified in XML, has been reached or you ask it to stop.

Multiple tasks can be started at the same time.

Click to start / stop requests from coming in.

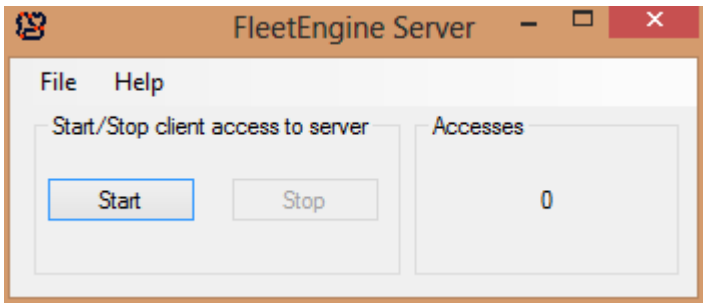
The server API is self-describing if you start and then call it like this:
http://localhost:1024/IRWFleetEngine?wsdl
http://localhost:1024/IRWFleetEngine

If you have UAC enabled, it is required to have elevated rights in order to start requests.

Methods

OptimizeStart: To start an optimization
GetStatus: To get status of a running optimization
GetStatusAll is used by the monitoring 18 application, but can also be used in your own applications.
GetResult: To get result of a running optimization
OptimizeStop: To stop a running optimization. On next call to GetResult, the result is returned and the task deleted.

Check out the C# sample application 19 for the exact work flow.



Windows NT Service

It can also be run as a Windows NT service. See Binary folder for batch files for installing / starting / stopping / uninstalling it as a Windows NT service.

When running as a service, it automatically starts.

Please remember, that services only have access to local drives, so don't setup networks in the INI file on LAN drives.

2.2.1 Configuration file

FleetEngine is being controlled by a configuration file, which is located in the same directory as the executable (fleetengine.exe or fleetengine.dll) and holds the information about which networks FleetEngine should load into memory and other settings.

Filename is the same as the executable, i.e. fleetengine.ini or fleetenginedesktop.ini. If you rename the executable, you will have to rename the ini file too.

0 / 1 settings always mean false / true.

[Server]

- Address: This need to be specified, if you make calls to the server directly from clients on other computers. Default is localhost.

- CPUAffinityMask: Controls which cores the server is allowed to use. Default 0 = all provided by OS. The numeric mask is a bit mask where each core is identified by a bit: bit 0 (value 1) identifies first core, bit 1 identifies second core etc. A mask of 6 (= 110 binary) thus means that core 2 and core 3 can be used by the server. Windows reports it as CPU's even though it is really cores.

- Directory specifies which directory to use when connecting to the server.

- MaxIdleTime specifies after how many seconds a task is deleted from the server, if it has not been queried. Default is 3600 sec = 1 hour.

- MaxRunningTasks specifies the maximum number of tasks that can be started at one time. Default is 10.

- Port specifies which port the server uses for communication with the client. Default is 1024.

[Optimizer]

- WarningOnInfeasibleJobs: By setting this to 1, you will get a warning if some of the jobs do not match with any of the resources. Otherwise (0) the optimization stops with an error. Default = 0.

- InitialSolutionGenerators: This can be used to specify which of the 4 initial solution generators should be used. By default all 4 are enabled and the best result is used for further improvements. You should only change this, if you want to get an initial solution in shorter time.

[Log]

- ChunkSize: The size (in bytes) of the chunks that the log file will be split into. Default is -1 = no splitting.

- Filename: Has a default value of application name + ".log" as extension. Alternative is a specific filename. You can also use environment variables such as %TEMP% for the temporary folder.

- LoggingLevel: Possible values are 0 - no logging, 1 - normal logging and 2 - debug logging (MANY details).

- Path: Points to the folder where the log file is stored. Default value is directory of executable / COM dll.

If the log file cannot be created in a service application for any reason, a windows event log record is created instead. Do not start two applications that write to the same log file. It won't work. This error can easily be made, when the configuration file is shared between different versions of FleetEngine.

[Network]
- Netmax defines the number of street networks defined in the next sections. If Netmax=3, the next sections should be "Net1", "Net2" and "Net3" (default 0).

[DrivingDirections] - Overview 23
- TemplateString = {Roadname} for {Dist} km, then {Turn} onto
- StraightOn = straight on
- SlightLeft = slight turn to the left
- Left = turn to the left
- SharpLeft = sharp turn to the left
- UTurn = make a u-turn
- SharpRight = sharp turn to the right
- Right = turn to the right
- SlightRight = slight turn to the right
- TakeExit = take {Exit} exit from roundabout

Settings for each network

[NetX]
- Coord3: 0 / 1 for loading coord3.bin & coord3i.bin file into RAM, 0 for reading it from disk. Use 1 if you have enough RAM (default 0).

- CoordinateWindow: This describes the allowed range for coordinates as input to functions. The number is an extra percentage around MBR (minimum bounding rectangle) of the street network (default 20). Use a negative value if you want to allow all values (not recommended).

- EncryptionKey: Set if encryption was used during network creation (default 0)

- ExternalID: 0 / 1 / 2 / 3 according to how and if external ID's should be loaded (default 0 = not loaded). 1 is sufficient for all normal purposes. 2 or 3 will make it cache more details in RAM, but without improving speed much. In any case this only applies to the network loading process.

- IgnoreNoDriveThrough: 0 / 1 according to if NoDriveThrough information should be ignored (default 0).

- IgnoreOneWay: 0 / 1 according to if one-way street information should be ignored. If you use a network for pedestrians you may prefer not to load one-way street information (default 0).

- Limit1: 0, 1 or 2. If 0, no limit files are loaded. If 1, limits defined by limit1.bin is loaded into memory as a scalar. If 2, limits defined by limit1.bin is loaded into memory as a bit pattern (default 0). See RW Net 4 documentation for details on limits.

- Limit2: Same as limit1, just for limit2.bin.

- Limit3: Same as limit1, just for limit3.bin.

- Path: Points to the folder with the network files. Required.

- Roadname: Allows you to open a single set of roadname files for output of driving directions with Viewer 24. Default value is 0.

- SpeedX: X is a value from 0 to 31, which is a road class. Defines the speed for that road class. If your network has road classes for which no speed information is defined, the default value is 60 km/h.

- Time: 0 / 1 according to if it should look for a file called *specialtime.txt* in the same directory as the other network files, which contains information about time information for specific links. See details here 18 (default 0).

- Turn: 0 / 1 according to if it should work in turnmode and look for a file called *turn.txt* in the same directory as the other network files, which contains information on turn restrictions in RW Net 4 format (default 0).

- UTurnAllowed: 0 / 1. Defined if U-turns are allowed or not (default 0 = not allowed). This refers to U-turns while driving from one job / depot to another. If you want to prevent U-turns at a job location, look at the Pickup 30 node in the XML.

2.2.1.1 Special time on links

You can define special travel time on single links by creating a file called "specialtime.txt" a standard text file, which holds one link ID and a travel time (in minutes) on each line. Use space as a separator.

Example:
12 5.8
100 15

Link 12 now has a travel time of 5.8 minutes, while link 100 has a travel time of 15 minutes.

This is typically used for ferries, car trains and other links, where the travel time isn't defined by the travel speed, but from timetable data. If you are using shortest path, this of course hasn't any effect since you cannot override the length of a link.

2.2.2 Monitor

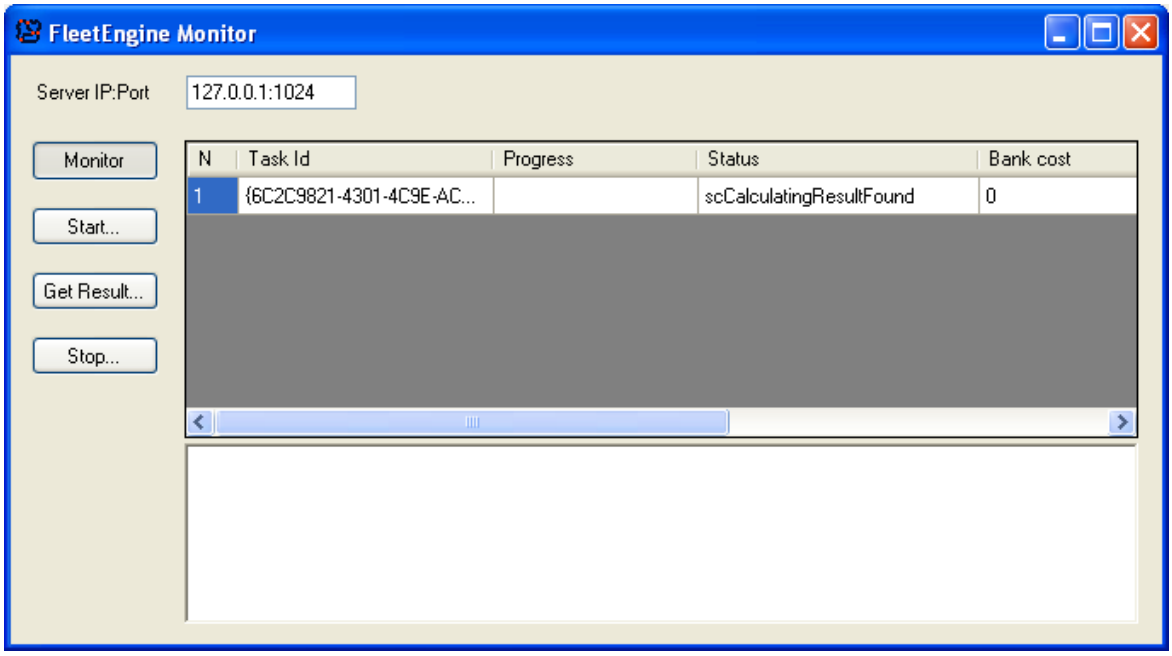
This application allows you to keep track of running tasks at the server, but also start / stop single tasks.

The monitor button can be used for toggling monitoring on / off.

Use the "Start" button to select a new XML input file and start a task.

Select a task from the list and then use the "Get Result" button to get the current result of the task, without stopping it. Status need to be scCalculatingResultFound. This will allow you to save the result in output XML format.

Use the "Stop" button to do as above, but also stopping the task on the server. If you click stop and then "cancel", the task will stop and the result is lost.

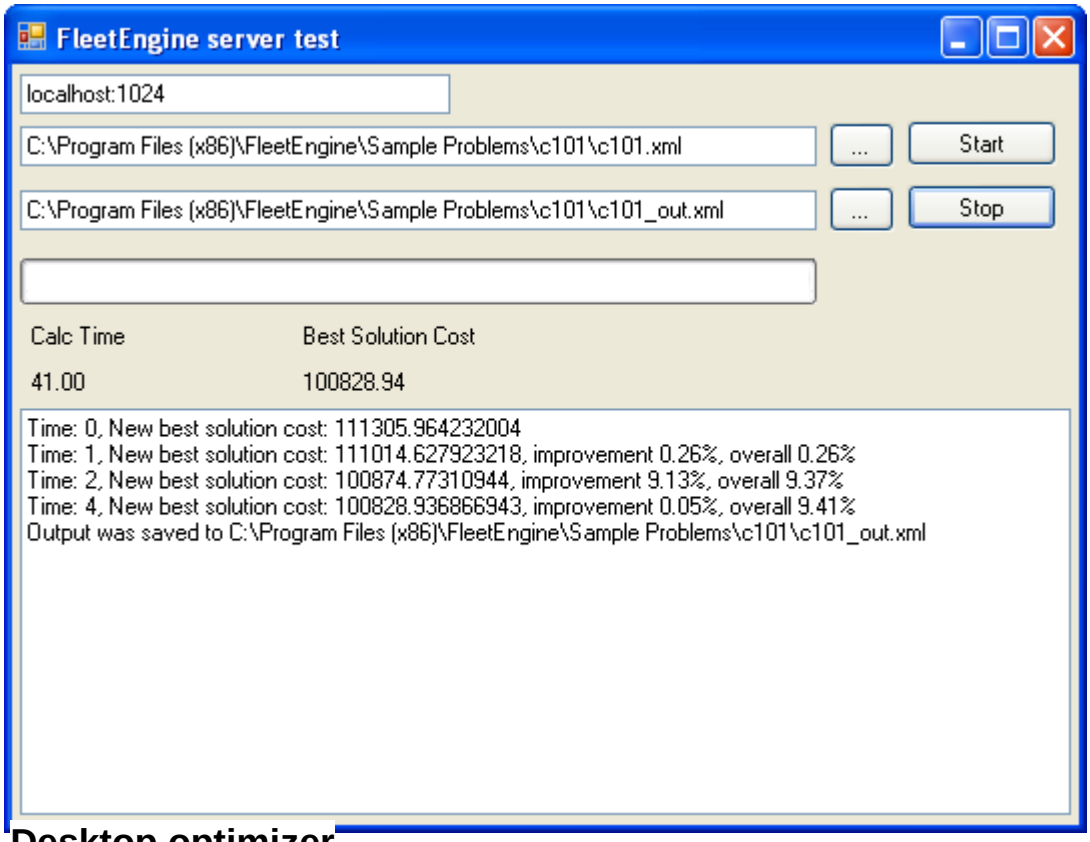


2.2.3 Sample

We have included a basic C# application that allows you to test the server.

It is fairly simple and allows you to start a task and then stop it again, unless the maximum calculation time has been reached in the meantime.

If you start multiple copies of the application, each can connect to the same server and run tasks.



2.3 Desktop optimizer

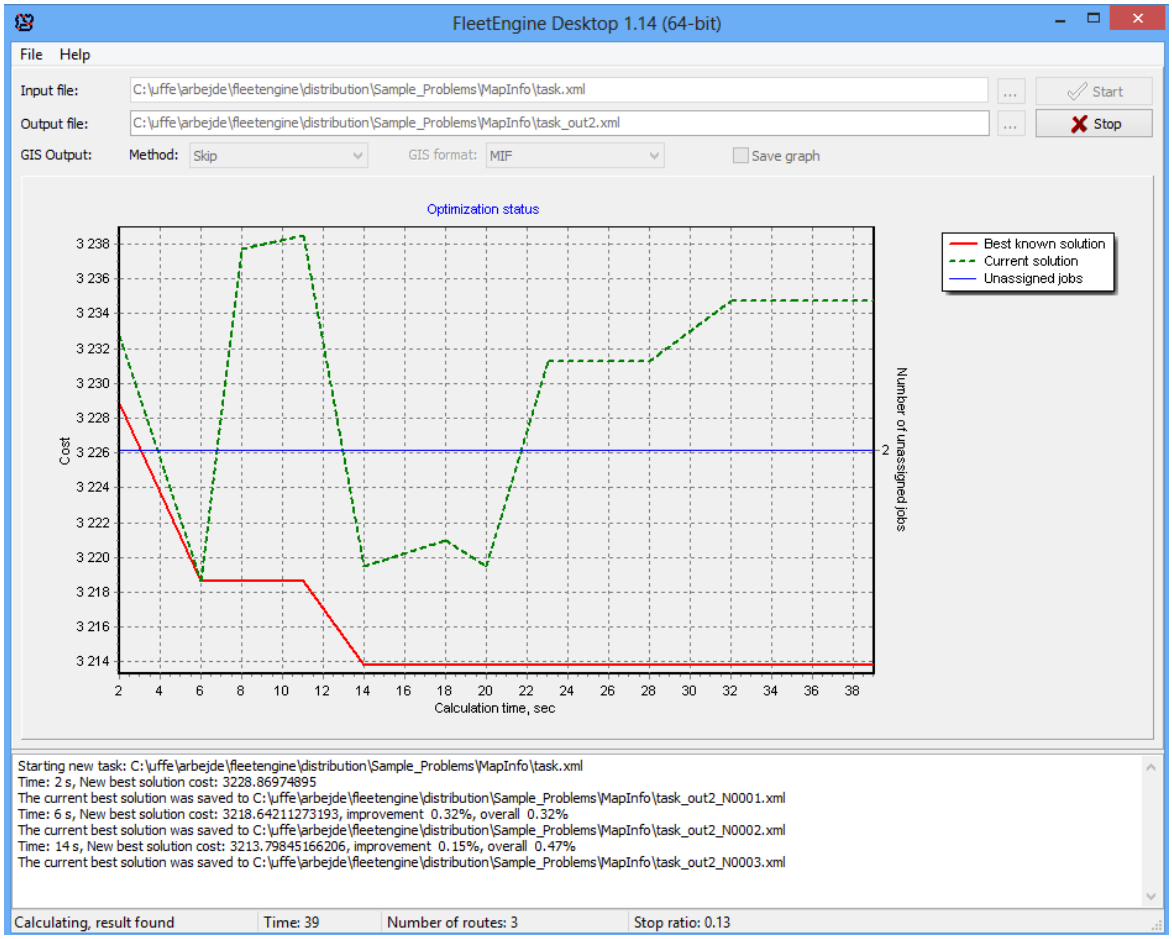
The Desktop optimizer allows you to run optimizations without starting the server or using the COM version:

- It can handle a single task at a time
- It runs locally
- Monitor is built-in and results presented on a graph
- XML output is nicely formatted, making it easier to view.
- It creates intermediate XML results automatically as new and better solutions are found.
- Graph can be saved in PNG format at the end of the process. Filename is the same as the input-file, just with PNG as extension.
- It can also call the Viewer 24 library automatically during the optimization process so you can create GIS files and follow output on a map.

Otherwise they share XML input/output 25 format, configuration file 16 format and the core optimizer. Both are self-contained.

No installation is needed, beyond setting up configuration file in advance. Just run FleetEngineDesktop.exe file from Binary folder.

Navigate to an input file, eventually update the output file name and finally click Start.



2.3.1 Batch run

It is possible to do batch runs with the Desktop version.

Parameters are either supplied through the command-line or the INI file.

- Command-line have preference:

See the batchrun.bat file for an example or call FleetEngineDesktop /h to see parameters.

- If done through the INI file, this is how it should be done:

[BatchMode]
Input = <filename>
Output = <filename>
GISformat = **MIF,** DBF, SHP, CSV, MITAB, GML2, KML2
GISmethod = **skip**, final, all
SaveGraph = **0**, 1
Quit = **0**, 1

Default values are adding "_out" to the output file name, MIF, skip and no.

2.4 COM library

The COM library is similar to the SOAP server except for a few differences:

- The interface is different (COM instead of SOAP)
- It can only handle a single task at a time (task ID is missing from all function calls)
- It only works locally
- It can not be monitored
- It can only be instantiated once, unless logging is turned off.

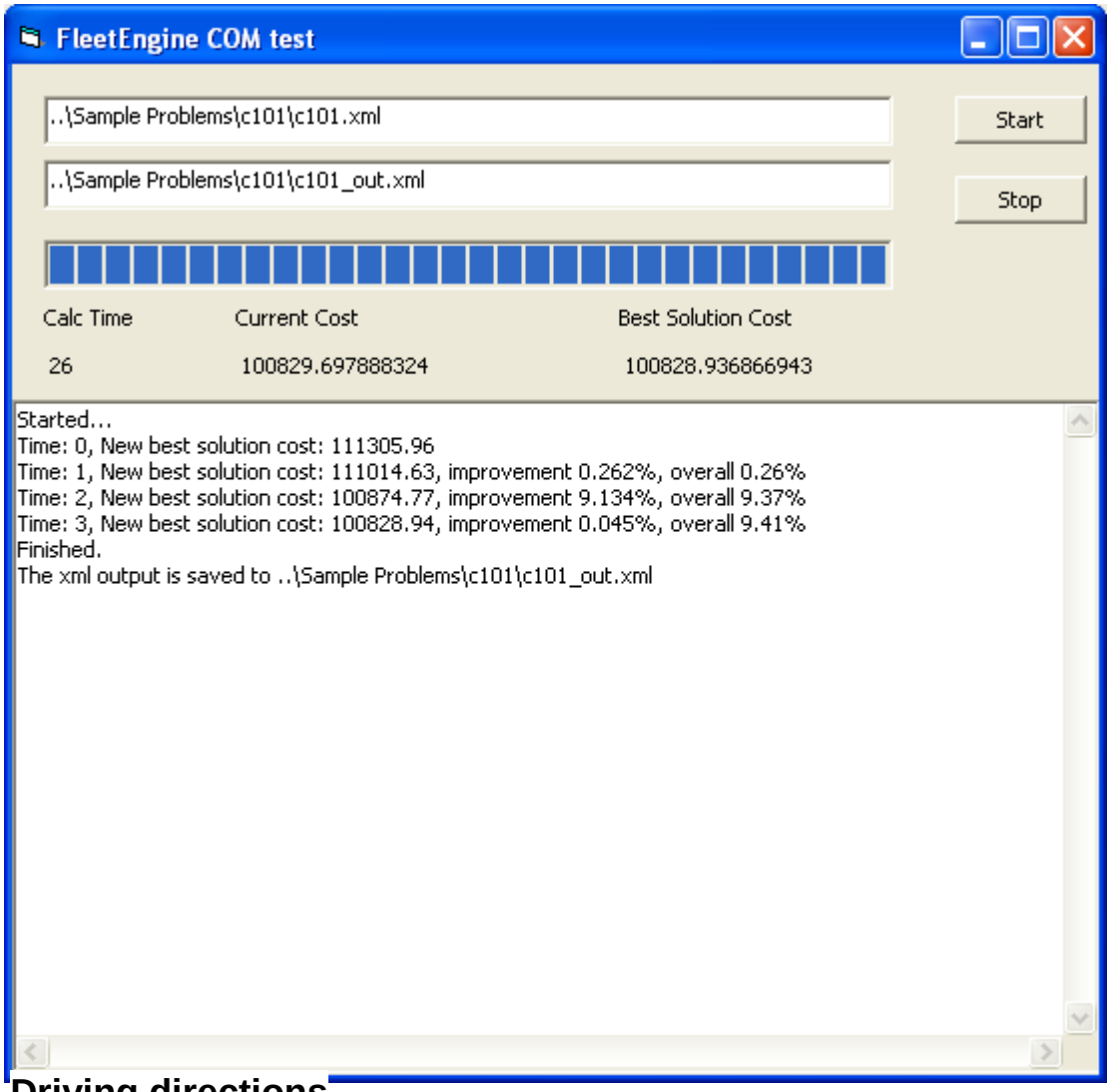
Otherwise they share XML input/output 25 format, configuration file 16 and the core optimizer. Both are self-contained.

Install it by running register_com.bat file in Binary folder.

2.4.1 Sample

We have included a Visual Basic 6.0 application that allows you to test the COM library.

It is fairly simple and allows you to start a task and then stop it again, unless the maximum calculation time has been reached in the meantime.



2.5 Driving directions

Driving directions in FleetEngine are a part of the normal XML output.

It only requires a street network as basis for the optimization and a field set up for road names.

An example output can be seen here (or in the TIGER sample output):

```
<GISRecords>
  <GISRecord>
    <Directions>
      <RoadName>Main Road</RoadName>
      <Distance>0.0829960251924088</Distance>
      <Time>0.0711394480361874</Time>
      <Turn>0</Turn>
      <Text>Main Road for 0.083 km, then onto</Text>
    </Directions>
    <Coordinates>
    ....
    </Coordinates>
  </GISRecord>
</GISRecords>
```

The content of <Text> can be customized via ini file settings, TemplateString. See the supplied INI file for the default setup.
Elements within {} will be replaced with corresponding values when creating the text. Non ascii characters are supported.

The value of <Turn> is used for creating the <Text> field and can generally be ignored by users.

The default language is English.

Viewer application processes this a bit further by summing distances and times.

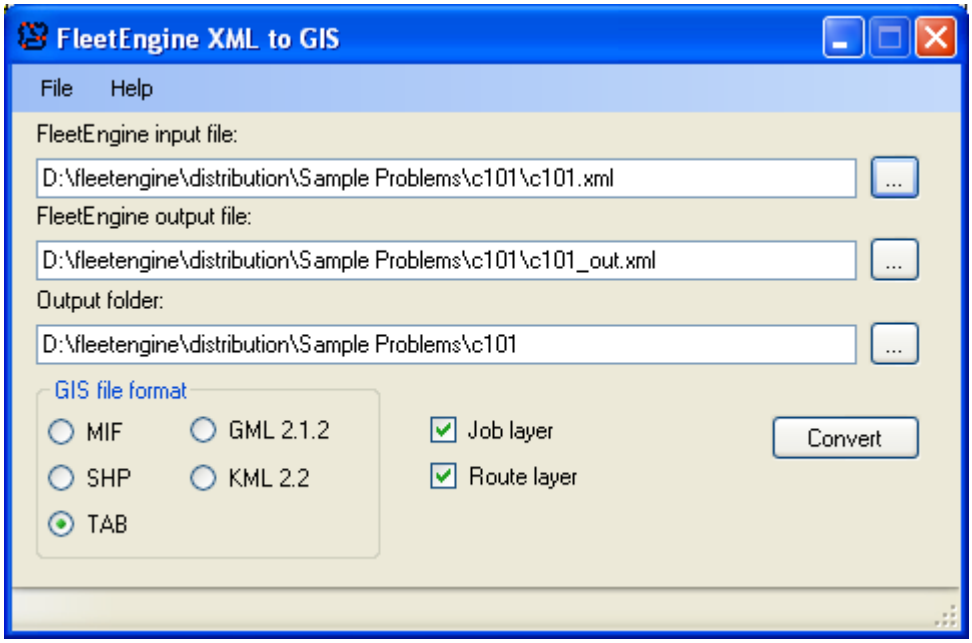
2.6 Viewer

The viewer application allows you to easily link FleetEngine input and output and save it in a number of popular GIS formats. It requires that you have set parameter 26 GISOutput before running FleetEngine.

File format can be chosen: TAB, SHP, MIF, KML or GML.

You need an actual GIS application to see the result. Several free ones are available: TatukGIS Viewer, ArcGIS Explorer, MapInfo ProViewer and many more.

If you choose KML as output format, you can use Google Earth for viewing. Please remember KML only works with geographic coordinates.



FeXmlConverterLib

FeXmlConverterLib is a .NET class library assembly which provides functionality to automate conversion of FE XML file to GIS tables, so you can add the same functionality as in the viewer to your own applications.

It exports class FeXmlConverter with two methods:

- void CreateGIS(bool aCreateJobsTable, bool aCreateRoutesTable, string aInputXmlPath, string

aOutputXmlPath, string aOutputPath, TGISFormat gisFormat)

Works with xml files specified by aInputXmlPath and aOutputXmlPath

- void CreateGISFromString(bool aCreateJobsTable, bool aCreateRoutesTable, string aInputXml, string aOutputXml, string aOutputPath, TGISFormat gisFormat)

Works with xml strings contained in aInputXml and aOutputXml

2.7 Schemas

The input and output file formats for FleetEngine are defined through two XML schemas.

We have also included the schemas in PDF format, for easier viewing.

Time is always minutes and distance is km. Cost is any unit you prefer.

2.7.1 Input XML

This chapter describes the input xml. Please look into the input.xsd file and samples for details. Especially the sample called dummy.xml shows all the *possible* elements of FleetEngine input.

You can refer to the xsd file inside the xml and it will automatically be validated when calling OPTIMIZESTART. We strongly recommend doing so. It will catch many possible errors as fast as possible.

But even so, we also recommend always testing input files for being valid with an XML validator, for instance XMLspy, before sending them to the optimizer.

If you don't have a validator, you can also use this website, which allow you to upload the schema too: www.xmlvalidation.com

When defining a task, use as few elements as possible for describing the problem to solve.
1) Only include resource types, that are actually in use.
2) Only use job time windows, which are smaller than the resource periods.
etc.

Task setup

4 main kinds of tasks are supported and depending upon this, input data about the jobs are organized differently.

- On-site - nothing is moved around and capacity is not relevant

The pickup 30 node is used for specifying the job.

- Pickup at specific locations and implicit delivery at the end of the route (at a depot if defined by the ResourcePeriod – else at the last job)

Typically waste collection. Just specify the coordinates of waste as the pickup 30 node.
Delivery is then automatically done at the end of the route.

- Implicit pickup at start of route (at a depot if defined by the ResourcePeriod – else at the first job) and delivery at specific locations

The classical delivery task (newspapers etc.). This can be modelled the same way as above, but using the delivery node only.

- Pickup at specific locations and delivery at specific locations

This can for instance be school children being picked up at home and brought to different schools. Specify both pickup and delivery node.
It can also be several items being moved from one location to other ones, where the vehicle returns to the pickup location to restock.
This mode is also known as "mixed pickup and delivery".

The advantage of only specifying the pickup or delivery node, is it allows the optimizer to match a job with the nearest depot, if there are multiple to choose between and you have no specific wishes, about which one it should be.

To prevent any confusion: A job always consists of a pickup and delivery action, which go together in pairs and are defined as one element in the XML, so there is no risk that you pickup more items than you deliver or vice-versa.

2.7.1.1 Parameters

Matrix refer to the source of distance/time matrix:
InXML: Provided as part of XML file 33
StreetNetwork: Calculated using street network(s)
StraightLineLatLong: Calculated using great circle distances (lat/long coordinates)
StraightLineMeter: Calculated using Pythagoras (meter-based coordinates)
StraightLineFeet: Calculated using Pythagoras (feet-based coordinates)

If Matrix is "StreetNetwork", FleetEngine uses dynamic segmentation and can also handle avoidance of u-turns (see Job 30 section) and calculate different matrices for different resource types. Coordinates are verified against the network setup - invalid coordinates returns an error.

Even when using InXML, it is still required to specify coordinates for depots and jobs, because we are using them in the optimization and the viewer 24 also uses these for generating a map of the result.

NumCapacities refer to the number of capacities used for the resources(default 0, maximum 5). If you have both a weight and count capacity, you have 2 capacities for instance.

SpeedKM can be set if Matrix is "StraightLine*". This is the speed used for translating the calculated distance into time. For obvious reasons the speed is the same everywhere. Unit is km/h, (default 60).

MaxCalculationTime is the maximum calculation time in minutes (Default 30). Calculations may stop sooner or you can stop it yourself, by calling the OPTIMIZESTOP method. Calculation time as reported by the server in function calls / events / desktop version is in seconds, while it is also minutes in the output XML. This only includes optimization time, matrix calculation time is on top of this.

MinCalculationTime is the minimum calculation time in minutes, to be used in combination with StopRatio (default 0).

StopRatio can be used as a generic stopping criteria. What it generally does is measuring the area under the red line in the graph in the desktop version 20. This is compared to the full area of the graph. When the ratio drops under the threshold and minCalculationTime has passed, it stops. With a value of 0.05, it will run for relatively long time before stopping. A higher value such 0.10 or 0.15 will make it stop after shorter time, if you want a quick solution (default 0 = disabled).

Area is measured from the last time, when the cost increased - typically because additional jobs got assigned. Similar minCalculationTime counts from last increase in cost.

RandomSeed allows you to replicate a previous run, by ensuring the same sequence of random numbers is used. By default a random number is used. This only works with the same version of the software.

GISOutput allows you to define if and how coordinates of the routes should be included in the output:
None: Skip (default)
Line: As straight lines between jobs / depots. This is usually preferred for the testing phase.
Route: As detailed routes following the street network. This makes the XML files a lot bigger.
LineRoute: As Lines in intermediate results and Routes in the final result.
Directions: As driving directions 23 (turn right, left etc).
LineDirections: As Lines in intermediate results and Directions in the final result.

Route / LineRoute / Directions / LineDirections are only allowed in combination with Matrix = StreetNetwork.

Use of Directions require that roadnames are loaded together with the street network in the INI file.

If you want to use the Viewer 24 application, GISOutput need to be different from None.

PriorityWeights
Normally job priorities work by preferring one job of higher priority rather than all jobs of lower priority.
If there is a choice. This is called "strict" and means weights are calculated automatically.

As an option you can define your own set of priority weights, so it isn't so strict.
Meaning for instance N jobs of lower priority may get preferred to 1 of higher priority.
You should define weights for as many priority levels as you use in the task.

1 is highest priority and 5 is lowest priority.

Priority also applies to standby jobs 31.

WorkloadBalancing
This feature allows you to keep load between different resources balanced, so the workload is more or less the same for everybody. This always comes at a cost. Assume the optimum solution is to let resource A execute 10 jobs and resource B execute 1 job. If you then transfer some jobs to resource B, the total cost may increase from increased driving distance.

By calculating the spread between workloads (standard deviation) and using a factor, it is possible to add an additional cost element to the total. This makes FleetEngine prefer solutions with a more even load between resources.

See supplied spreadsheet workload_balancing.xls for an example.

Requires a minimum of 2 resources.

2.7.1.2 Depots

Depots may be home addresses of drivers (resources), a garage or the current location (GPS-based) of a driver. Generally where routes should start or end.

A depot is defined by an ID and a set of coordinates.

A depot is not necessarily where items are being picked up or delivered to, unless it happens to be the same coordinates.

2.7.1.3 Competences

All competences should be listed here.

For each resource type you can specify its competences.
For each job a list of competences (requirements) should be listed.

In order for a resource type to match a job, all competences for the job needs to be listed for the resource type.

If all resource types matches all jobs, no competences has to be defined.

Example for a fleet of vehicles
Resource type 1: Has competences Red and Luxury
Resource type 2: Has competences Red
Resource type 3: Has competences Black and Air-conditioned

Job1 requires Red: Match with type 1 and 2.
Job2 requires Luxury: Match with type 1.
Job3 requires Red and Air-conditioned: No match (and warning in output XML).

2.7.1.4 Resource type

This is the main element describing a resource, the part that carries out a job:

A resource type is defined by an ID.

These variables identifies the cost associated with using the resource type:

Cost: This is cost if the resource 29 is allocated to any job. This needs to be >0, if you want to reduce the number of resources as much as possible. Larger vehicles will typically have a higher cost than smaller vehicles.
In a typical case where costdrivetime=1, use a value that reflects the length of a workday such as 500.

CostKM: Cost per km
CostDriveTime: Cost per minute (driving)

If costKM = 0, then fastest route is used for route calculations when Matrix = StreetNetwork.
If costDriveTime = 0, then shortest route is used for route calculations when Matrix = StreetNetwork.
Otherwise a weighted cost is used in route calculations.
They can not both be 0.

CostWaitTime: Cost per minute (waiting). This is normally a lower value than CostDriveTime.

CostServiceTime: Cost per minute (during job pickup and job delivery). This should only be set if multiple resource types are in use and they have different CostWaitTime. In that case, use the same value as CostWaitTime.

CostPerGroupChange: Cost per group change 33
CostPerSubRoute: Cost per sub route 34.

CostLoadKm: Cost per load*km - defined for each load index. This is for situations where the cost depends upon the load - higher load leading to higher driving costs.

See Service Time 35 for description of **ServiceTimePerJob** and **ServiceTimePerDemand**.

By setting **Capacity** it is possible to define the maximum load at any time during a resource period. Capacity can refer to number of seats, maximum weight, volume or similar physical restriction. This is the counterpart to demand for jobs 30. The sum of demands for jobs assigned to a resource period, must not exceed capacity.

By setting **AccumulatedCapacity** it is possible to define the maximum sum of demands for all pickups within a resource period.

The difference is load (capacity) increases or decreases as items are being picked up or delivered, while AccumulatedLoad (AccumulatedCapacity) only increases when something is being picked up. It is only in the case of mixed pickup and delivery there is a difference between the two.

Competences of the resource type can also be defined.

You may want to define **MaxTime** if there is a maximum time allowed for a resource type, even if a resource period is longer. Similarly **MaxDist** and **MaxCost** can be defined. Default is 0, meaning not in use.

UTurnTime can be defined to prevent too many u-turns. A typical value could be 0.5 (minute).

Additional options when *Matrix* = "StreetNetwork":

Network refers to the network to be used. Networks are defined in the configuration file 16.

If you are using multiple networks for the same task, make sure the coordinate system is the same for all networks and all jobs & depots are covered by the networks.

Limits refers to up to 3 limits.

A limit could be the weight of a vehicle, to prevent it from going anywhere where the network do not allow it. Limit is a number from 0 to 255. Default is 0 = no limit.

2.7.1.5 Resource

You can define one or more resources. A resource is the actual instantiation of resource types. If you for instance have 10 vehicles of a specific type, then you should create 10 entries in the input XML.

A resource is defined by an ID and always refers to one of the **resource types**.

StartLate
If you are working with job timewindows, it may sometimes be possible to adjust the starting time from the depot, so the start is later, but still keeping servicetime within the time windows. Sometimes this may even reduce waiting time. This is the default behaviour.

By setting this parameter to false, a resource period always starts as the resource period defines it, possibly adding waiting time.

WorkloadHistory
It is possible to supply a history of workloads, so the balancing can take the past into account: If resource A worked a lot last week, he/she may not have to work so much this week to keep the total in balance. See supplied spreadsheet workload_balancing.xls for an example.

Resource periods
For each resource it can be defined, when it is available as one or more resource periods.

A period is defined by an ID. This is included in the output, but do not have to be unique.

A period will usually start and stop at a depot, but doesn't have to. If any of the depots are not defined, we call it an open-ended task.

Depots can be different: **Startdepot** can for instance be current location for the resource, while **stopdepot** is home address.

Finally the key parameters **Start** and **Stop**. These may even be different days, since it is defined as a timestamp.

Breaks
You can define multiple breaks per resource period. They are defined by a **Start** and **Stop**, which is the earlist time the break can start, and the latest it can end. **Length** of break (in minutes) need to fit within the period.

Break length does not affect the cost function, as it's not a part of total service time.

Breaks have higher priority than jobs, meaning that breaks are always allocated before jobs.

If breaks are located between two locations with a long drivetime between, the drivetime is split in the output. Otherwise breaks are normally located just before or after a job.

Breaks cannot split a job service time. This means if you have a break from 10.00 to 10.30, you can not be assigned a one hour job from 9.30 to 11.00.

2.7.1.6 Jobs

This is the actual job to be matched with a resource period. See here 25 for details about how to setup jobs.

Competences of the jobs are a list of all the competences 28 required by the resource type.

Demands can be defined, similar to Capacity for resource types 28 (default 0).

Priority is a number from 1 to 5 (1 is highest priority).

Preferred Resources can be used to define, which resource(s) should be used to carry out a job.
If penaltycost>0, the value is added to the total cost, if the resource is not on the list.
If strict is true, only the listed resources can be used and penaltycost is ignored.

2.7.1.6.1 Pickup

Either Pickup or Delivery 31 node is required.

Jobs should have a set of coordinates (**X,Y**).

ServiceTime is the time it requires to execute the pickup (default 0). Also see Service Time 35.

TimeWindows can be defined for the job. It is simply a start and stop timestamp.
It means the job has to start on or after the start timestamp and similarly finish before or on the stop timestamp.
Which also means the servicetime needs to fit within the window.
If more than one is defined, it means any *one* of the time windows can be used.
If the time windows has different costs, it will try to use the one with the smallest cost, all other factors considered too.
Default is no time windows and no cost.

Precedence can be used to make sure certain jobs are being executed before other jobs. This

applies within a resource period, so unless two jobs are assigned to the same resource period, this has no effect. Low precedence is executed before high precedence (default 0). Pickup for a job is always before delivery no matter precedence.

Additional options when Matrix 26 = StreetNetwork:

AvoidUTurn can be specified to make sure no U-turns are made (default false) at the pickup / delivery point.

If set to true for jobs on cul-de-sac links, it will be changed to false since a U-turn is needed anyway. A warning is included in the output when this occurs.

Direction can be used to control curb approach. In some situations it is required that a resource approaches the job on a particular side of the street. 5 values are possible:
-1: Any direction is accepted (default).
0: Direction should be in the digitized direction of the street.
1: Opposite of digitized direction.
2: Correct side of road in countries with right-hand driving
3: Correct side of road in countries with left-hand driving (UK, Australia, New Zealand, India etc)

If AvoidUTurn or Direction is set for any job, calculation of distance- and time matrix is done accordingly (takes more time).

2.7.1.6.2 Delivery

Either Pickup 30 or Delivery node is required.

The delivery node is the same as the pickup 30 node, except there is an additional entry:

MaxOnBoardTime (default no limit) is the maximum time from pickup to delivery point.

MaxOnBoardTime in combination with a delivery timewindow do not work at the moment. It should be avoided.

Example
In school bus routing a maximum of X minutes inside the vehicle is allowed. This can be different for each job, if younger students are not allowed inside the bus for so long time as older students.

2.7.1.7 Standby jobs

Compared to a normal job, a standby job is one without a specific location. It is meant for situations where a resource need to be ready in case of emergencies. If you have a standby job with a specific location, use normal jobs instead.

Priority is a number from 1 to 5.

Competences of the jobs are a list of all the competences required by the resource type.

TimeWindow defines when the standby job should be. It is required, compared to normal jobs.

2.7.1.8 Job Groups

Job groups are various ways to describe how groups of jobs interact: Match / do not match / need to be assigned pairwise etc.

More such restrictions will be added by time.

2.7.1.8.1 Incompatible jobs

This allows you to define groups of jobs which are not compatible with all other jobs. But compatible with each other.

This is similar to how sheeps and lions do not go together at the same time (the lion is likely to eat the sheep).

Example:

Jobs: J1, J2, J3, J4 and J5.
If J1 and J2 are listed as incompatible, it means they can not be loaded onto a resource at the same time as J3, J4 and J5.

OK, since J1 is deliveried before J3 is picked up:
J1 - pickup
J1 - delivery
J3 - pickup
J3 - delivery

Not OK, since J1 and J3 is loaded onto the resource at the same time.
J1 - pickup
J3 - pickup
J1 - delivery
J3 - delivery

2.7.1.8.2 Job Interval

This allows you to define that 2 jobs should be in a specific order:

<First> and then <Second>. No matter which resource period, they are assigned to.

Furthermore it can be specified how many minutes they should be apart.
You can define an optimal level and minimum / maximum.

If you define cost, the optimizer will try to get closer to optimal. Cost is measured per minute and is usually a fairly small number.

If cost is 0, there is no need to specify optimum.

This can be used to plan for visiting a customer every N days. An example with twice a year:

Job1: Use time window = Jan - June
Job2: Use time window = July - December

First = Job1
Second = Job2
MinInterval = 182 * 24 * 60 * (0.9) = 235872
MaxInterval = 182 * 24 * 60 * (1.1) = 288288
OptimalInterval = 182 * 24 * 60 = 262080
Cost = 0.01 (or 14.4 units for every day)

This keeps the 2 visits 6 months +/- 10% apart.

2.7.1.9 Matrix

If matrix parameter is "InXML", you need to supply a (distance, time) matrix as part of the input XML. Please see the sample (dummy.xml) for exact format.

Number of elements in matrix: (Depots+2*Jobs)*(Depots+2*Jobs-1)

InXML only allows one resourcetype, no job directions and no standby jobs. If no delivery node in the XML, use 0 in the string below.

Order of depots and jobs determine the order in the (distance, time) matrix. Example with 2 depots and 2 jobs:

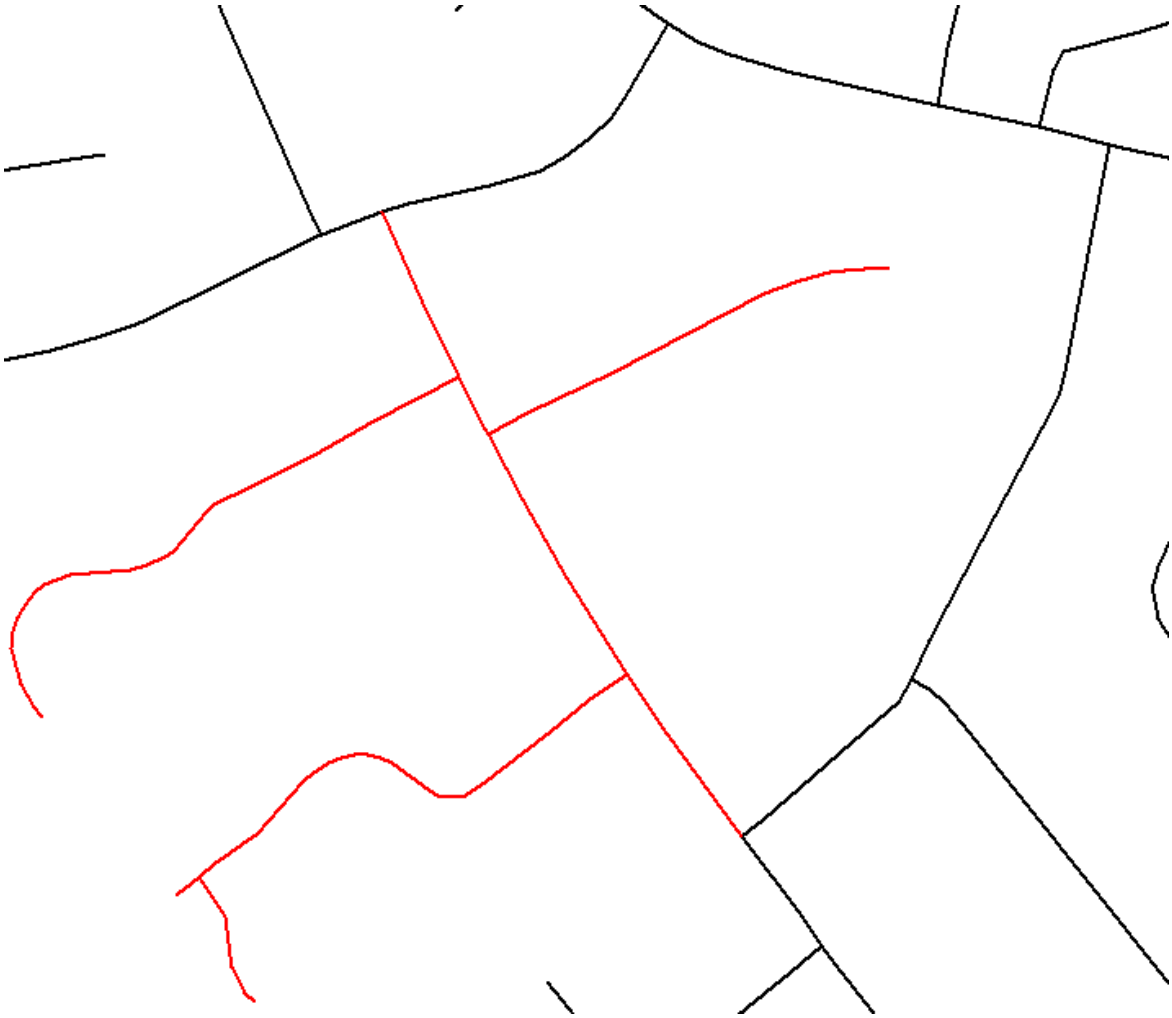
Depot1 > Depot2
Depot1 > Job1_Pickup
Depot1 > Job2_Pickup
Depot1 > Job1_Delivery
Depot1 > Job2_Delivery
Depot2 > Depot1
Depot2 > Job1_Pickup
Depot2 > Job2_Pickup
Depot2 > Job1_Delivery
Depot2 > Job2_Delivery
Job1_Pickup > Depot1
Job1_Pickup > Depot2
Job1_Pickup > Job2_Pickup
Job1_Pickup > Job1_Delivery
Job1_Pickup > Job2_Delivery
Job2_Pickup > Depot1
Job2_Pickup > Depot2
Job2_Pickup > Job1_Pickup
Job2_Pickup > Job1_Delivery
Job2_Pickup > Job2_Delivery
Job1_Delivery > Depot1
Job1_Delivery > Depot2
Job1_Delivery > Job1_Pickup
Job1_Delivery > Job2_Pickup
Job1_Delivery > Job2_Delivery
Job2_Delivery > Depot1
Job2_Delivery > Depot2
Job2_Delivery > Job1_Pickup
Job2_Delivery > Job2_Pickup
Job2_Delivery > Job1_Delivery

2.7.1.10 Group

In tasks with densely grouped jobs, it may be desired to keep these together, so jobs are assigned just after each other in order to generate more "natural" routes. Even if this may be suboptimal due to time or capacity constraints.

FleetEngine can automatically locate such groups. On the map below the red links show a single group and all jobs on these links belong to the same group. By setting a cost on change of groups, they will automatically be kept together during optimization. The exception are jobs which have direction set to 0 or 1.

Groups are defined as all links between 2 intersections with a node degree > 2. Links that are part of a cul-de-sac are not included in the calculation of node degree, but they are part of the group. This means the 4 links plus corresponding cul-de-sac in the example below:



2.7.1.11 Sub route

A sub route is typically relevant for routes with many pickups or deliveries where it is needed to go to a depot several times to get emptied or refilled, depending upon task and capacity. This means most jobs either has the same pickup or delivery coordinates.

Number of sub routes are simply counted as the number of times a pickup job is followed by a delivery job.

Example with 2 sub routes:
Home depot
Pickup1
Pickup2
Pickup3
Pickup4
Delivery1 - Delivery2 - Delivery3 - Delivery4 (empty vehicle)
Pickup5
Pickup6
Pickup7
Delivery5 - Delivery6 - Delivery7 (empty vehicle)
Home depot

Example with 3 sub routes:
Home depot
Pickup1 - Pickup2 (fill vehicle)
Delivery1
Delivery2
Pickup3 - Pickup4 (fill vehicle)
Delivery3
Delivery4
Pickup5 - Pickup6 - Pickup7 (fill vehicle)
Delivery5
Delivery6
Delivery7
Home depot

2.7.1.12 Service time

Service time is normally specified is a Job 30 parameter, but can be specified as a resource specific constant too, if different resources types have different characteristics.

ServiceTime
This is the base service time, specified for the job itself. Most setups will only use this one.

ServiceTimePerJob
This is a resource type specific constant.

ServiceTimePerDemand
This value depends on the "size" of the job , where a value for each capacity index can be defined.

Example
When picking up students there may be small and big busses (resource types 28), where the small busses can make a stop in shorter time, but the students can get in and out faster from the big bus (multiple doors).

Small bus, pickup:
ServiceTimePerJob = 0.4 (24 sec)
ServiceTimePerDemand = 0.10 (6 sec)

5 students: 0.4 + 5 x 0.10 = 0.90 min (54 sec)

Big bus, pickup:
ServiceTimePerJob = 0.6 (36 secs)
ServiceTimePerDemand = 0.05 (3 sec)

5 students: 0.6 + 5 x 0.05 = 0.85 min (51 sec)

Similar values can be entered for delivery.

2.7.2 Output XML

This chapter describes the output xml. Please look into the output.xsd file and samples for details.

XML files smaller then 1 MB are nicely formatted for easier reading, when using the Desktop version.

- Build (identifies exactly which version of FleetEngine you are using)
- Calculation time

- LastImprovementTime
- DistanceMatrixCalculationTime
- Stop reason
- Random seed (for possible re-use in input parameters 26)
- Cost
- Dist
- GroupChanges
- SubRoutes
- Service time, Drive time, Total time, Wait time

These are all the totals for the whole solution.

Errors
These are critical situations that prevent further optimizations:

- Network node for resource type is defined and no corresponding network in configuration file 16
- Non-connected elements in matrices
- Error in XML input
- Cannot find competence
- Cannot find depot
- Cannot find resource type
- Break and breaklength nodes should be both defined or both undefined
- Break is not between start and stop
- Break length does not fit between start and stop
- Start is later than stop
- Length of the time interval is zero
- x2 and y2 should be both defined or both undefined
- Destination is unreachable (closed link)
- Task restrictions are too tight
- Cannot find job assignment
- Cannot calculate distance
- Cannot find route from
- Cannot create initial solution
- Invalid number of servicetimeperload elements
- Invalid number of capacity elements
- Invalid number of load elements
- Speed cannot be 0
- Job cannot be served by any ResourceType
- Network index is incorrect
- Loads element does not correspond to the number of capacities
- Capacities element does not correspond to the number of capacities
- ServiceTimePerLoads element does not correspond to the number of capacities
- Processing stopped
- Infeasible model
- Invalid coordinates
- Network was not loaded
- User stop before result was found
- License file is incorrect or not found
- License limits are exceeded
- Feature is not supported in this version of FleetEngine
- Unexpected error

Warnings

A warning is when something doesn't look right in the input, but not critical enough to stop optimizations:

- A resource type 28 cannot serve any jobs.
- A job cannot be assigned to any resource period (meaning it will always end as un-assigned).
- Best known solution is infeasible
- Network node is ignored when Matrix is not equal to StreetNetwork
- Limits node is ignored when Matrix is not equal to StreetNetwork
- Direction is ignored when Matrix is not equal to StreetNetwork
- SpeedKM node is ignored
- Matrix data is ignored

Resource periods & Jobs
Main output includes the same information twice:

- As a list of resource periods, start/stop depot and the order of jobs within the period. This includes all information about drive time/distance between jobs, waiting time, service time and eventually direction in/out (curb approach).
- As a list of all jobs and when they are executed. Starttime is when the resource arrives at the job, then waiting time occurs and finally job execution begins.

Un-assigned jobs
These are included for easier reference.

2.8 RW Net 4 version

This is a version that only allows a single ResourcePeriod in optimizations. Everything else is the same as normal FleetEngine.

These features are supported:

- Breaks
- Job priorities
- Standby jobs
- Time windows
- Capacity

Consequently these features are not relevant (but technically supported):

- Multiple resource types
- Competences
- Workload balancing
- More than 2 depots (one for start and one for stop)

You can use this version under the same license terms as you use the rest of RW Net 4 Pro.

Part III

Sample tasks

3 Sample tasks

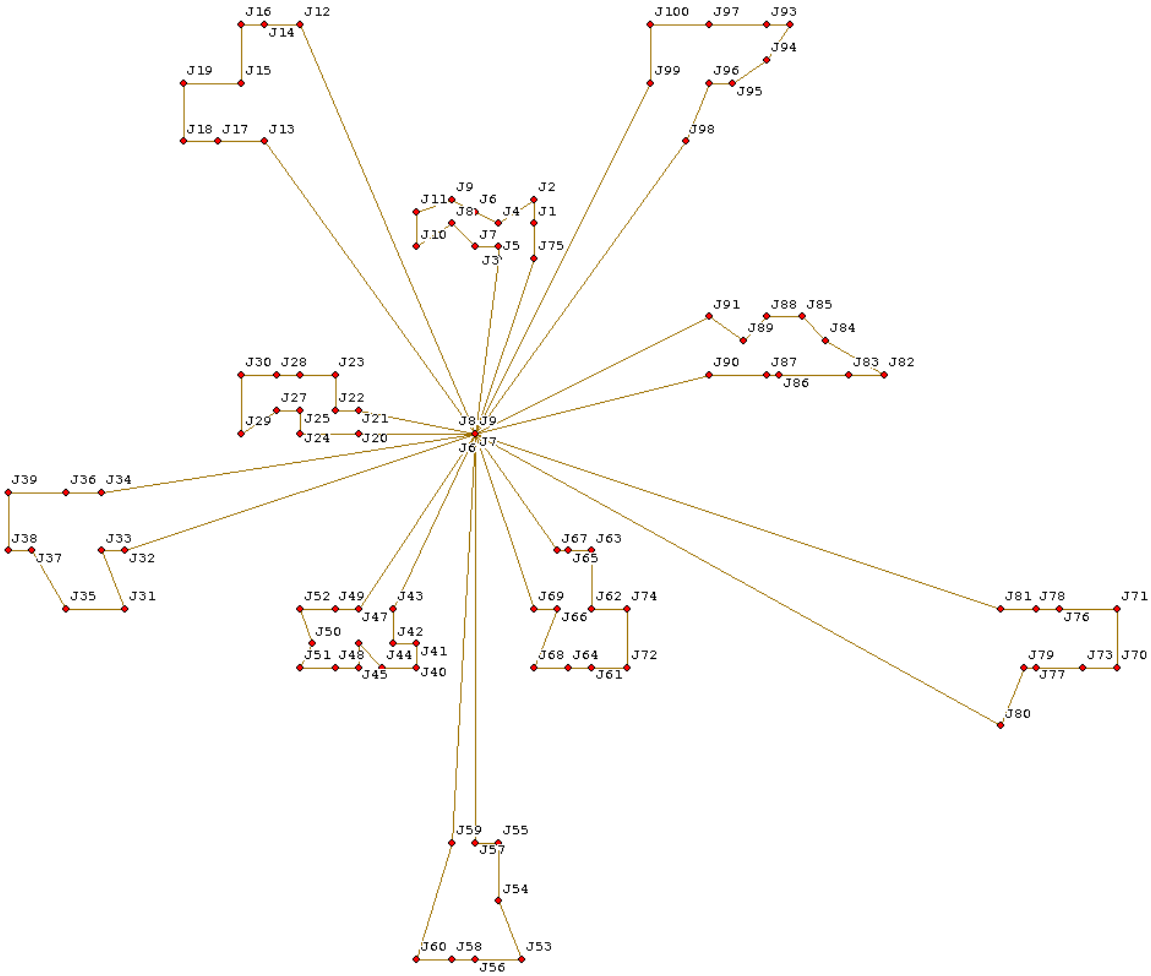
You will also find a simple XML file, which illustrates most of the elements in input XML for easier understanding of the syntax. See dummy.xml. This is not suited for actual optimization.

- If you want to use FleetEngine for simple A to B routing, that is possible too. See two samples on how to do it.
- A more thorough sample based around OSM 42 and a setup in ArcGIS & MapInfo applications are included. These show how to generate XML from a database and running it through FleetEngine and the viewer.
- We have also included C101 41 from the Solomon test cases.

3.1 C101

This is a the first dataset in the collection of Solomon datasets. This is a standard benchmark dataset within VRP research, originally created by Marius M. Solomon. It uses coordinates and straight line distances. It includes 100 jobs with different time windows and 10 vehicles to carry out the task.

It gets solved to optimality:

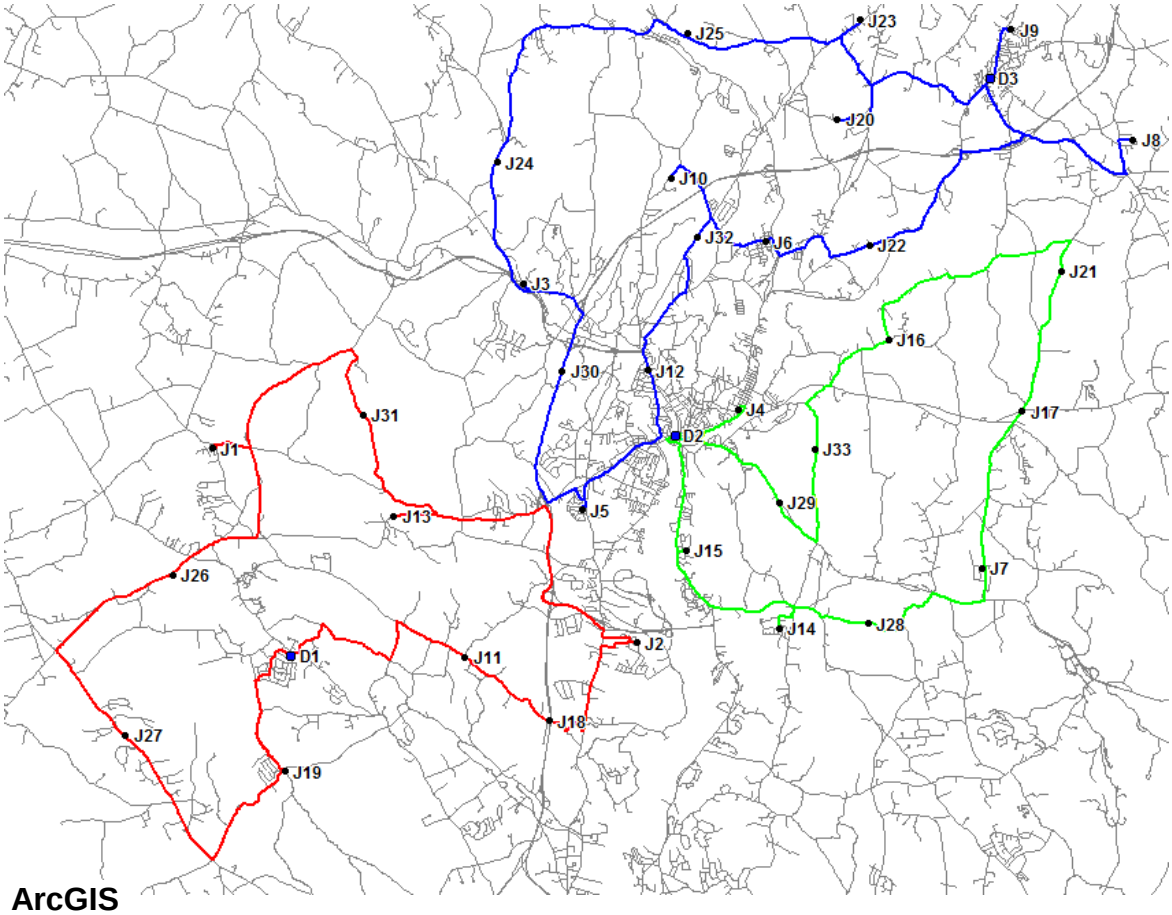


3.2 OSM

We have included a small sample based around the OpenStreetMap street network (supplied too) from the area of Connecticut, USA.

It includes 3 depots (D1-D2-D3), 1 resource type, 3 resources (one at each depot) and 33 jobs with varying service time. The resource at D2 has shorter working hours (5h) than the other 2 depots (8h and 10h), which leads to routes of different lengths.

The background data are supplied in ArcGIS 43 and MapInfo 44 format, so you can generate the FleetEngine XML file on your own.



3.2.1 ArcGIS

This is an application for turning an ArcGIS file geodatabase into FleetEngine XML format, using Python.

The data are the same as those in the OSM 42 sample and uses the same network.

Generally it is meant as a starting point for your own development / experiments.

The model doesn't include all possible elements in FleetEngine:
Job time windows and direction are for instance excluded, but can be added.
Similar it shows pickup jobs only.

Steps
Open the MXD (to load the required data). This requires that link.shp is in folder \network\shp\ as in the default setup.
Open the python window, right click and load "FleetEngine_demo.py".
Press the Carriage Return button when in the Python window to run the program.
XML file is written to the directory containing the Python program.

You can now run the generated task.xml file through FleetEngine.

Other options for running Python code
Python code can be called from ArcGIS modelbuilder, added as a tool in Toolbox (and placed behind a button on a toolbar) or called from a .NET Arcobjects application. ESRI provides the ArcPy site package – this can be accessed by an IDE outside of the ArcGIS python window – such as Eclipse.

System requirements
ArcGIS version 10.0 (Basic / Standard / Advanced)
Python 2.6 (for use of ArcPy site package installed with ArcGIS 10)

Pre-Requisites
Familiarity with ArcGIS File Geodatabase format (.GDB)
Basic understanding of ArcGIS 10 ArcPy site package and Python 2.6

3.2.2 MapInfo

This is an application for turning a number of TAB files into FleetEngine XML format, using MapBasic.

The data are the same as those in the OSM 42 sample and uses the same network.

Generally it is meant as a starting point for your own development / experiments.

The model doesn't include all possible elements in FleetEngine:
Job time windows and direction are for instance excluded, but can be added.
Similar it shows pickup jobs only.

Steps
You should open map1.wor to open all needed files.
Run the mbx file.
This generates task.xml

You can now run this XML file through FleetEngine.

System requirements
The precompiled version requires MapInfo 10.5.
If you want to recompile, MapBasic 9.0 or later is required due to the use of datetime fields.

Pre-Requisites
Familiarity with MapInfo and running an MBX file.