Project PLATON

Planning Process and Tool for Step-by-Step Conversion of the Conventional or Mixed Bus Fleet to a 100% Electric Bus Fleet

Document type: Description of experimental software for solving problem OPT by randomized heuristic RH.

Editor: Nikolai Guschinsky (UIIP-NASB)

Contributors: Nikolai Guschinsky, Mikhail Y. Kovalyov, Boris Rozin (all from UIIP-NASB)

Grant beneficiary of WP leader: the United Institute of Informatics Problems of the National Academy of Sciences of Belarus

Funding organization of WP leader: National Academy of Sciences of Belarus, Independence Ave. 66, 220072, Republic of Belarus, Minsk

Abstract

This document contains description of an experimental software for solving optimization problem OPTSCHED. The problem and algorithm SY are described in Deliverables 4.3 and 4.4. The problem OPTSCHED is to distribute departures of buses of the same type assigned to the same route as uniform as possible over the departures of all buses serving this route in the decisive time period. The timetables which address this objective are called balanced. It is assumed that buses of different types have different passenger capacities. Therefore, a balanced timetable ensures a uniform allocation of bus capacities over time in the decisive time period of the same route.

Contents

1 Computer implementation of algorithm SY

Algorithm SY is implemented in C++ for Windows. It can be used as an executable file *schedule.exe* or as a DLL-file *scheddll.dll*. These files can be used on a PC of a standard configuration. Parameters of the command line for *schedule.exe* are:

- Full name of directory with input data.
- Full name of directory with configuration file *sched.ini*.

For example: *d:/gn/soft/mobility/schedule/schedule.exe d:/gn/soft/mobility/mobility/schedule d:/gn/soft/mobility/schedule*, where *d:/gn/soft/mobility/schedule* is the directory with *schedule.exe*, *d:/gn/soft/mobility/schedule* is the directory with the input data, and *d:/gn/soft/mobility/schedule* is the directory with the configuration file *sched.ini*.
From Python *schedule.exe* can be executed in the following way:

```
import subprocess
argexe='d:/gn/soft/mobility/schedule/schedule.exe'
arg1=' d:/gn/soft/mobility/schedule'
arg2=' d:/gn/soft/mobility/schedule'
args = argexe + arg1+ arg2
p=subprocess.Popen(args, shell = False)
p.wait()
ret=p.poll()
```

File *scheddll.dll* contains function *SCHED*, whose prototype is *int SCHED(char * dir,char * dir_ini)*, where *dir* is the full name of the directory with the input data and *dir_ini* is the full name of the directory with the configuration file *sched.ini*. The return code of the function *SCHED* is equal to 0 if the optimization was successful. In this case, all the output information is placed into the file *sched.out* in the text format and in the file *sched.json* in the JSON format in the directory *dir*. If the return code is not 0, then the corresponding error information is placed into the file *errors.out* in the directory *dir*. An example of calling the function *SCHED* from Python (32-bit) is given below.

```
import ctypes
schDll=ctypes.WinDLL("d:/gn/soft/mobility/schedule/scheddll.dll")
from ctypes import *
p1=create_string_buffer(b"d:/gn/soft/mobility/schedule")
p2=create_string_buffer(b"d:/gn/soft/mobility/schedule")
ret=mobDll.SCHED(p1,p2)
```

File *sched.ini* is used for setting parameter *json*:

- $json$ – format of the input data, $json \in \{0, 1\}$, where

  $json = 0$ if the input data are in the text format,

  $json = 1$ if the input data are in the JSON format .

  $json = 1$ is the default value.

Now GUI application (file *schedulev.exe*) is available. The application supports viewing and printing of optimization results.

2 Formats of input file

Two formats of the input file are implemented. One of them is the JSON format, see *http://www.json.org/index.html* for the description, and the other is the simple text format. If the input parameter $json = 1$, then the file *schedin.json* is converted into the text file *schedin.txt*. The data from the text file are imported and analyzed for errors. If there are errors, then the information about them is placed into the file *errors.out* in the directory specified by the parameter *dir*.
The file *schedin.json* defines values for names *sn_b* (short names of the e-bus and conventional vehicle types) and *nv_b* (number of vehicles of type *b*) are to be defined. For example:
{
sn_b: [*MAZ103, E433*],
nv_b: [*3, 8*]
}
The file *schedin.txt* consists of two rows. The first row includes short names of the e-bus and conventional vehicle types. The second row contains number of vehicles of each type. For example:
MAZ103, E433
3, 8

3 Formats of the output file

Two formats of the output file are implemented. One of them is the JSON format, and the second is the simple text format.
Object *t* is defined in the file *sched.json*. It consists of the short names of e-bus and conventional vehicle types. For example:
{*t:* [
E433, MAZ103, E433, E433, E433, MAZ103, E433, E433, E433, MAZ103, E433]
}

The obtained solution, which is the sequence of vehicle types in the departure order, is placed into the file *sched.out*. It consists of the sequence of short names of e-bus and conventional vehicle types. For example:

E433 MAZ103 E433 E433 E433 MAZ103 E433 E433 E433 MAZ103 E433