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 DepOpt.

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 DepOpt. The problem and the solution method are described in Deliverables 4.3 and 4.4. Problem DepOpt deals with a single depot and a set of e-buses with slow charging batteries, which are repetitively charged at this depot and serve a given set of trips each. The problem is to determine the required electric power supplied to the depot by the city power grid, types and number of charging stations in the depot, types of e-bus batteries and charging times of each e-bus while it is in the depot such that the total daily cost of charging equipment, e-bus batteries and consumed energy is minimized, provided that the arrival and departure times of e-buses to/from the depot, the dynamic upper bound on the supplied power and functions of charge and discharge of the batteries are addressed.

Contents

1 Computer implementation of DepOpt solution method

The method for solving DepOpt is implemented in C++ for Windows. It can be used as an executable file *charge.exe* or as a DLL-file *chargedll.dll*. These files can be used on a PC of a standard configuration. In both cases, DLL-library *lpsolvevcdll.dll* is used for calling Open Source *Mixed Integer Linear (MIP)* Solver LpSolve. This library has to be placed into the same directory as *charge.exe* and/or *chargedll.dll*.
Parameters of the command line for *charge.exe* are:

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

For example: *d:/gn/soft/mobility/charge/charge.exe d:/gn/soft/mobility/charge/rozin1 d:/gn/soft/mobility/charge*, where *d:/gn/soft/mobility/charge* is the directory with *charge.exe*, *d:/gn/soft/mobility/charge/rozin1* is the directory with the input data, and *d:/gn/soft/mobility/charge* is the directory with the configuration file *charge.ini*.
From Python, *charge.exe* can be executed in the following way:

```
import subprocess
argexe='d:/gn/soft/mobility/charge/charge.exe'
arg1=' d:/gn/soft/mobility/charge/rozin1'
arg2=' d:/gn/soft/mobility/charge'
args = argexe + arg1+ arg2
p=subprocess.Popen(args, shell = False)
p.wait()
ret=p.poll()
```

File *chargedll.dll* contains function *CHARGE*, whose prototype is *int CHARGE(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 *probl.ini*. The return code of the function *CHARGE* is equal to 0 if the optimization was successful. In this case, all the output information is placed into the file *solution.out* in the text format, and in the file *solution.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 *CHARGE* from Python (32-bit) is given below.

```
import ctypes
chargeDll=ctypes.WinDLL("d:/gn/soft/mobility/charge/chargedll.dll")
from ctypes import *
p1=create_string_buffer(b"d:/gn/soft/mobility/charge/rozin1")
```

p2=create_string_buffer(b"d:/gn/soft/mobility/charge")
ret=chargeDll.CHARGE(p1,p2)
File *charge.ini* is used for setting the following parameters:

- $json$ – format of the input data, $json \in \{0, 1, 2\}$, where

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

  $json = 1$ if the input data are in the JSON format in separate files,

  $json = 2$ if the input data are in the JSON format in one file.

  $json = 2$ is the default value.

- $hour$ – format of departure and arrival times of trips, $hour \in \{0, 1\}$, where

  $hour = 0$ if the times are given in format "hour.min",

  $hour = 1$ if the times are given in format "hour".

  For example: 5.30 if $hour = 0$ and 5.5 if $hour = 1$.

  $hour = 0$ is the default value.

- typ_pr – type of execution, $typ\_pr \in \{1, 3, 4, 5\}$, where

  $typ\_pr = 1$ if the optimization should be executed for all types of batteries,

  $typ\_pr = 3$ if only transformation of input text files to separate json files should be executed,

  $typ\_pr = 4$ if only transformation of input text files to one json file should be executed,

  $typ\_pr = 5$ if the optimization should be executed for set of batteries specified by parameter $batteries$,

  $typ\_pr = 1$ is the default value.

- $batteries$ – list of short names of batteries selected for installation on e-buses.

- n_max_var – maximum number of subproblems $\mathbf{B}_2(p_D, c, \mathbf{b})$ (to be solved for a fixed upper bound p_D of the supplied power and the same type c of the charging stations, see Deliverable 4.3). If the actual number of subproblems $\mathbf{B}_2(p_D, c, \mathbf{b})$ is greater than n_max_var, then the *Particle Swarm Optimization (*PSO*)* method is used instead of the full enumeration, and $n\_max\_var = 10000$ is the default value.

- nit – maximum number of iterations of PSO, $nit = 1000$ is the default value.

- nit_non – maximum number of iterations of PSO without improving the value of the objective function, $nit\_non = 100$ is the default value.

- npt – number of particles of the PSO method, $npt = 100$ is the default value.
- w – control parameter ω of the PSO method, $w = 0.729$ is the default value.
- fiP – control parameter φ_p of the PSO method, $fiP = 1.49445$ is the default value.
- fiG – control parameter φ_g of the PSO method, $fiP = 1.49445$ is the default value.

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

2 Formats of input files

Two formats of the input files are implemented. One of them is the JSON format, see *http://www.json.org/index.html* for the description, and the other is a simple text format. If the input parameter $json = 2$, then file *charge.json* is transformed into the following files: *stations.json, batteries.json, fbatteries.json, fcbatteries.json, buses.json, routes.json, trips.json, fleet.json, febuses.json, fpower.json.* Then, each of these files is converted into the corresponding text file. Finally, the data from the text files are imported and analysed 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*.

2.1 JSON format

If the input data are prepared in the JSON format in separated files, then their names must be the following: *stations.json, batteries.json, fbatteries.json, fcbatteries.json, buses.json, routes.json, trips.json, fleet.json, febuses.json, fpower.json.*
File *stations.json* describes set C of charging stations types and defines values of the following parameters: *fn_c* (full name of the charging station), *sn_c* (short name of the charging station), *po_c* (nominal power po_c),*cc_cap_c* (capital cost cc_c^{cap}) and *cc_ope_c* (operating and depreciation cost cc_c^{ope}).

```
{ "C": [{
"fn_c": "Charging station",
"sn_c": "CS",
"po_c": 200,
"cc_cap_c": 8590,
"cc_ope_c": 5000
}
]}
```

File *batteries.json* describes set B of batteries types and defines values of the following parameters: *fn_bat* (full name of the battery), *sn_bat* (short name of battery), *smin_bat* (minimal SOC level),*smax_bat* (maximal SOC level), *costb_bat* (cost), *C_bat* (short names of the eligible charging stations), and *chr_bat* (charging rates of the eligible charging stations). For example:
{ "B": [{
"fn_bat": "Battery 1",
"sn_bat": "B1",
"smin_bat": 94.05,
"smax_bat": 470.25,
"costb_bat": 202206,
"C_bat": ["CS"],
"chr_bat": [188.1]
}
]}
The following parameters are defined in the file *fbatteries.json*: *sn_bat* (short name of the battery b) and functions $f_{bc}(\tau)$ for each eligible charging station c from C_b. Functions are defined by *a_1, b_1, k1_1, k2_1, . . . , a_n, b_n,k1_n,k2_n* where n is the number of eligible charging stations ($|C_b|$). Here *a_i* is the array of left endpoints of the segments of the function for ith charging station from C_b, *b_i* is the array of the right endpoints of the segments of the function for ith charging station from C_b, *k1_i* is the array of the coefficients $k1$ of the function for ith charging station from C_b, and *k2_i* is the array of the coefficients $k2$ of the function for ith charging station from C_b. For example:
{ "BF": [{
"sn_bat": "B1",
"a_1": [0],
"b_1": [2],
"k1_1": [188.1],
"k2_1": [94.05]
}
]}
The following parameters are defined in the file *fcbatteries.json*: *sn_bat* (short name of the battery) and function $N_b(b_b^{low})$. The function is defined by *soc_bat* (discharge levels in percents) and *ncycl_bat* (maximal numbers of charge/discharge cycles). For example:
{ "CBF": [{
"sn_bat": "B1",
"soc_bat": [10,20,30,40,50,60,70,80,90,100],
"ncycl_bat": [450000,150000,50000,24000,14000,7000,4400,3000,2300,1900]

}
]}
File *buses.json* describes the set EB e-buses types and defines: *fn_b* (full name of the e-bus type), *sn_c* (short name of the e-bus type), and *B_b* (short names of the eligible batteries). For example:
{ "EB": [{
"fn_b": "Vitovt Max Electro E433",
"sn_b": "E433",
"B_b": ["B1"]
}
]}
File *routes.json* describes the set R of routes and defines values of the parameters *fn_r* (full name of the route), *sn_r* (short name of the route), and *l_r* (the total length of the route). For example:
{ "R": [{
"fn_r": "Railway Station - DS Viasnjanka",
"sn_r": "A1",
"l_r": 9
}
]}
File *trips.json* describes the set TR of trips and defines values of the parameters *fn_t* (full name of the trip), *sn_t* (short name of the trip), *dep_t* (departure time), *arv_t* (arrival time) and *R_t* (short names of the routes). For example:
{ "T": [{
"fn_t": "Trip 1",
"sn_t": "T1",
"dep_t": 5.13,
"arv_t": 22.44,
"R_t": ["T43"]
}
]}
File *fleet.json* describes the set J of e-buses fleet and defines values of the parameters *fn_j* (full name of the e-bus), *sn_j* (short name of the e-bus), *sn_b_j* (short name of the e-bus type), *Nc_j* (annual number of charge/discharge cycles), *B_j* (short names of the eligible batteries) and *TR_j* (short names of the trips). For example:
{ "J": [{
"fn_j": "2816",

"sn_j": "2816",
"sn_b_j": "E433",
"Nc_j": 350,
"B_j": ["B1"],
"TR_j": ["T1"]
}
]}
The following parameters are defined in the file *febuses.json*: *sn_j* (short name of the e-bus) and functions $\varphi_{\mu bp}(s)$ for each pair (battery $b \in B_j$, trip $p \in TR_j$). The functions are defined by *a_1_1, b_1_1, k1_1_1, k2_1_1, ..., a_1_n2, b_1_n2, k1_1_n2, k2_1_n2,*a_2_1, b_2_1, k1_2_1, k2_2_1, ..., a_2_n2, b_2_n2, k1_2_n2, k2_2_n2,...,a_n1_n2, b_n1_n2, k1_n1_n2, k2_n1_n2 where *n1* is the number of eligible batteries in B_j and *n1* is the number of served trips in TR_j. Here *a_i_l* is the array of the left endpoints of the segments of the function for ith battery from B_j and lth trip from TR_j, *b_i_l* is the array of the right endpoints of the segments of the function for ith battery from B_j and lth trip from TR_j, *k1_i_l* is the array of the coefficients $k1$ of the function for ith battery from B_j and lth trip from TR_j, and *k2_i_l* is the array of the coefficients $k2$ oof the function for ith battery from B_j and lth trip from TR_j. For example:
{ "FJ": [{
"sn_eb": "2816",
"a_1_1": [94.05],
"b_1_1": [470.25],
"k1_1_1": [1],
"k2_1_1": [-376.2]
}
]}
File *fpower.json* includes definitions of functions $P(t)$, $c_e(t)$ and $cost(p_D)$: *T_p* (endpoints of the periods for $P(t)$), *F_p* (shares of the power at the endpoints of the periods), *T_c* (endpoints of the periods for $c_e(t)$), *F_c* (tariffs at the endpoints of the periods), *Pd* (powers supplied), *Cpd* (costs of the power). For example:
{
"T_p": [5,12,20,24],
"F_p": [1,1,1,1],
"T_c": [5,12,20,24],
"F_c": [0.065,0.065,0.065,0.065],
"Pd": [200,400,600,800],
"Cpd": [2000,4000,6000,8000]
}

2.2 Text format of the input files

If the input data are prepared in the text format, then the following files must be created: *stations.txt, batteries.txt, fbatteries.txt, fcbatteries.txt, buses.txt, trips.txt, fleet.txt, febuses.txt, fpower.txt*. Each file can include comments. They must start with the symbols // and be placed at the top of the file. The main body of the file starts with a new line immediately after the comments. Values in the rows are separated by commas.

File *stations.txt* consists of one row for each element of the set C charging stations types. Each row contains: full name of the charging station, short name of the charging station, nominal power, capital cost, operating and depreciation cost. For example: *Charging station 1, CS1, 200, 250000, 5000*.

File *batteries.txt* consists of three rows for each element of the set B batteries types. The first row contains: full name of the battery, short name of the battery, minimal SOC level, maximal SOC level, cost of the battery. The second and third rows contain short names and charging rates of the eligible charging stations for the battery, respectively. For example, row 1: *Battery 1, B1, 94.05, 470.25, 202206*, row 2: *CS1*, row 3: *188.1*.

File *fbatteries.txt* consists of the data for each element of the set B. The first row contains short name of the battery. The next $4 \cdot |C_b|$ rows consist of the data for the functions $f_{bc}(\tau)$ for each eligible charging station c (first two rows specify left a and right b endpoints respectively of charging time segments, the last two rows define coefficients $k1$ and $k2$ of the linear functions for each of these segments). For example, row 1: *B1*, row 2: *0*, row 3: *2*, row 4: *188.1*, row 5: *94.05*.

File *fcbatteries.txt* consists of three rows for each element of the set B. The first row contains short name of the battery. The next two rows consist of the data for the function $N_b(b_b^{low})$ (first row specifies the discharge levels, second row defines maximal number of charge/discharge cycles). For example, for *Battery 1*, row 1: *B1*, row 2: *10,20,30,40,50,60,70,80,90,100*, row 3: *450000, 150000, 50000, 24000, 14000, 7000, 4400, 3000, 2300, 1900.*

File *buses.txt* consists of two rows for each element of the set EB of e-buses types. The first row consists of full and short names of the e-bus type. The second row contains short names of the eligible batteries. For example, for e-bus type *Vitovt Max Electro E433*, row 1: *Vitovt Max Electro E433, E433*, row 2: *B1*.

File *routes.txt* consists of one row for each element of the set R of trips in the format *full name of the route, short name of the route, total length of the route*. For example, for the route *DS Drugnaja - DS Siarova*: *DS Drugnaja - DS Siarova, T43, 7*.

File *trips.txt* consists of two rows for each element of the set TR of trips. The first row contains full name of the trip, short name of the trip, departure time and arrival time. The second row consists of the short names of the served routes. For example, row 1: *Trip 1, T1, 5.13, 22.44*, row 2: *T43*.

File *fleet.txt* consists of three rows for each element of the set J of e-buses fleet. The first row contains full name of the e-bus, short name of the e-bus, short name of the e-bus type, annual number of its battery charge/discharge cycles. The second row includes short names of the eligible batteries. The third row contains short names of the eligible trips. For example, row 1: *2816, 2816, E433, 350*, row 2: *B1*, row 3: *T1*.
File *febuses.txt* consists of the data for each element of the set J. The first row contains short name of the e-bus. The next rows include data for the functions $\varphi_{jbp}(s)$ (4 rows) for each pair (battery $b \in B_j$, trip $p \in TR_j$). It is assumed that trip is changed first. First two rows specify arrays a and b of left and right endpoints of its battery SOC level segments. The next two rows define arrays $k1$ and $k2$ of function coefficients). For example, row 1: *2816*, row 2: *94.05*, row 3: *470.25*, row 4: *1*, row 5: *-376.2*.
File *fpower.txt* consists of 6 rows and defines functions $P(t)$, $c_e(t)$ and $cost(p_D)$. The first two rows contain endpoints of the periods and shares of supplied power at the endpoints of these periods. The next two rows include endpoints of the periods and fixed power rates in these periods. The last two rows include powers supplied and costs of the power. For example, row 1: *5, 12, 20, 24*, row 2: *1, 0.85, 0.85, 1*, row 3: *5, 12, 20, 24*, row 4: *0.065, 0.065, 0.065, 0.065*, row 5: *200, 400, 600, 800*, row 6: *2000, 4000, 6000, 8000*.

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.

3.1 JSON format of the output file

Two objects are defined in file "solution.json": "UR" (general characteristics of the obtained solution) and "CHB" (parameters for each e-bus).
The object "UR" defines: "pd" (optimal value p_D^* of the power supplied to the depot), "c" (short name of the optimal charging station type c^*), "K" (number K^* of charging stations of the type c), "ttu" (total unused daily charging time resource $\bar{t}^{ch}$), "stu" (share Ψ^{ch} of the used daily resource of charging time in the depot , "cst" (optimal cost), "tu_i" (unused charging time resource in the time intervals), "ts_i" (beginning of time intervals with unused charging resource), "tf_i" (end of time intervals with unused charging resource). For example:
{"UR": {
"pd": 400,
"c": "CS",
"K": 1,
"ttu": "16:33",

″stu″: 0.69,
″cst″: 31948.6,
″tu_i″: [″0:14″,″0:11″,″0:33″,″6:00″,″0:29″,″2:14″,″4:26″,″2:10″,″0:07″,″0:10″],
″ts_i″: [″0:00″,″5:16″,″5:27″,″6:00″,″12:00″,″12:29″,″15:34″,″20:00″,″22:10″,″22:16″],
″tf_i″: [″5:00″,″5:27″,″6:00″,″12:00″,″12:29″,″15:34″,″20:00″,″22:10″,″22:16″,″22:26″]
}
Object ″CHB″ defines parameters: ″sn_eb″ (short name of e-bus), ″sn_bat″ (short name of used battery), ″ct_eb″ (the total charging time), ″SOC_d″ (SOC level for each departure), ″T_d″ (time moment for each departure), ″SOC_a″ (SOC level for each arrival), ″T_a″ (time moment for each arrival), ″ct_i″ (charging time for each charging), ″ts_i″ (start times of charging) and ″tf_i″ (finish times of charging). For example:
{
″CHB″: [
{
″sn_eb″: ″2816″,
″sn_bat″: ″B1″,
″ct_eb″: ″2:00″,
″SOC_d″: [470.25],
″T_d″: [″5:13″],
″SOC_a″: [94.0501],
″T_a″: [″22:44″],
″ct_i″: [″0:31″,″0:13″,″0:24″,″0:52″],
″ts_i″: [″0:00″,″5:00″,″22:44″,″23:08″],
″tf_i″: [″5:00″,″5:13″,″23:08″,″24:00″]
}]

3.2 Text format of the output file

The obtained solution is placed into the unique file *solution.out*. The output includes: optimal value of the power supplied to the depot, optimal type of the charging station and their number; output for each e-bus, unused charging time resource for each period, total unused daily charging time resource, share of the used daily resource of charging time, and the total cost.
For each e-bus, the output is:

- Selected battery type.
- Total charging time.
- Charging times and periods of charging.

4 Input files for Minsk case

In this section input files in JSON and text format are provided for Minsk case.

4.1 File *charge.json* for Minsk case

{
″stations″:
{ ″C″: [{
″fn_c″: ″Charging station″,
″sn_c″: ″CS″,
″po_c″: 200,
″cc_cap_c″: 8590,
″cc_ope_c″: 5000
}
]}
,
″batteries″:
{ ″B″: [{
″fn_bat″: ″Battery 1″,
″sn_bat″: ″B1″,
″smin_bat″: 94.05,
″smax_bat″: 470.25,
″costb_bat″: 202206,
″C_bat″: [″CS″],
″chr_bat″: [188.1]
}
]}
,
″fbatteries″:
{ ″BF″: [{
″sn_bat″: ″B1″,
″a_1″: [0],
″b_1″: [2],
″k1_1″: [188.1],
″k2_1″: [94.05]
}
]}
,
″fcbatteries″:
{ ″CBF″: [{
″sn_bat″: ″B1″,
″soc_bat″: [47.03,70.5,94.1,117.6,141.1,164.6,188.1,211.6,235.1,258.6,282.2,305.7,329.2,352.7,376.2,400,423.2
″ncycl_bat″: [450000,280000,150000,80000,50000,35000,24000,18000,14000,9200,7000,5600,4400,3500,3000,
}
]}
,
″buses″:
{ ″EB″: [{
″fn_b″: ″Vitovt Max Electro E433″,
″sn_b″: ″E433″,
″B_b″: [″B1″]
}
,
{

″fn_b″: ″Vitovt Electro E420″,
″sn_b″: ″E420″,
″B_b″: [″B1″]
}
]}
,
″routes″:
{ ″R″: [{
″fn_r″: ″Railway Station - DS Viasnjanka″,
″sn_r″: ″A1″,
″l_r″: 9
}
,
{
″fn_r″: ″Dolgobrodskaja - DS Siarova″,
″sn_r″: ″T59″,
″l_r″: 12.2
}
,
{
″fn_r″: ″DS Drugnaja - DS Siarova″,
″sn_r″: ″T43″,
″l_r″: 7
}
]}
,
″trips″:
{ ″T″: [{
″fn_t″: ″Trip 1″,
″sn_t″: ″T1″,
″dep_t″: 5.13,
″arv_t″: 22.44,
″R_t″: [″T43″]
}
,
{
″fn_t″: ″Trip 2″,
″sn_t″: ″T2″,
″dep_t″: 5.27,
″arv_t″: 12.48,
″R_t″: [″T43″]
}
,
{
″fn_t″: ″Trip 3″,
″sn_t″: ″T3″,
″dep_t″: 15.56,
″arv_t″: 22.16,
″R_t″: [″T43″]
}
,
{
″fn_t″: ″Trip 4″,
″sn_t″: ″T4″,
″dep_t″: 5.45,
″arv_t″: 23.08,

″R_t″: [″T43″]
}
,
{
″fn_t″: ″Trip 5″,
″sn_t″: ″T5″,
″dep_t″: 6,
″arv_t″: 22.27,
″R_t″: [″T43″]
}
]}
,
″fleet″:
{ ″EB″: [{
″fn_eb″: ″2816″,
″sn_eb″: ″2816″,
″sn_b″: ″E433″,
″Nc″: 350,
″B_eb″: [″B1″],
″T_eb″: [″T1″]
}
,
{
″fn_eb″: ″2818″,
″sn_eb″: ″2818″,
″sn_b″: ″E433″,
″Nc″: 700,
″B_eb″: [″B1″],
″T_eb″: [″T2″,″T3″]
}
,
{
″fn_eb″: ″2807″,
″sn_eb″: ″2807″,
″sn_b″: ″E433″,
″Nc″: 350,
″B_eb″: [″B1″],
″T_eb″: [″T4″]
}
,
{
″fn_eb″: ″2805″,
″sn_eb″: ″2805″,
″sn_b″: ″E433″,
″Nc″: 350,
″B_eb″: [″B1″],
″T_eb″: [″T5″]
}
]}
,
″febuses″:
{ ″EBF″: [{
″sn_eb″: ″2816″,
″a_1″: [94.05],
″b_1″: [470.25],

″k1_1″: [1],
″k2_1″: [-376.2]
}
,
{
″sn_eb″: ″2818″,
″a_1″: [94.05],
″b_1″: [470.25],
″k1_1″: [1],
″k2_1″: [-160.74],
″a_2″: [94.05],
″b_2″: [470.25],
″k1_2″: [1],
″k2_2″: [-136.8]
}
,
{
″sn_eb″: ″2807″,
″a_1″: [94.05],
″b_1″: [470.25],
″k1_1″: [1],
″k2_1″: [-376.2]
}
,
{
″sn_eb″: ″2805″,
″a_1″: [94.05],
″b_1″: [470.25],
″k1_1″: [1],
″k2_1″: [-352.26]
}
]}
,
″fpower″:
{
″T_p″: [5,12,20,24],
″F_p″: [1,1,1,1],
″T_c″: [5,12,20,24],
″F_c″: [0.065,0.065,0.065,0.065],
″Pd″: [200,400,600,800],
″Cpd″: [2000,4000,6000,8000]
}
}

4.2 Input text files for Minsk case

4.2.1 File ″stations.txt″

Charging station 1,CS1,200,250000,5000
Charging station 2,CS2,200,250000,5000
Charging station 3,CS3,200,250000,5000

4.2.2 File "batteries.txt"

Battery 1,B1,94.05,470.25,202206 What is this?
CS1,CS2,CS3 What is this?
Battery 2,B2,94.05,470.25,202206
CS1,CS3
Battery 3,B3,94.05,470.25,202206
CS2,CS3

4.2.3 File "fbatteries.txt"

B1
0
2
188.1
94.05
0
2
188.1
94.05
0
2
188.1
94.05
B2
0
2
188.1
94.05
0
2
188.1
94.05
B3
0
2
188.1
94.05
0
2
188.1
94.05

4.2.4 File "fcbatteries.txt"

B1
47.03,70.5,94.1,117.6,141.1,164.6,188.1,211.6,235.1,258.6,282.2,305.7,329.2,352.7,376.2,400,423.2,446.7,470.2
450000,280000,150000,80000,50000,35000,24000,18000,14000,9200,7000,5600,4400,3500,3000,2600,2300,210
B2
47.03,70.5,94.1,117.6,141.1,164.6,188.1,211.6,235.1,258.6,282.2,305.7,329.2,352.7,376.2,400,423.2,446.7,470.2
450000,280000,150000,80000,50000,35000,24000,18000,14000,9200,7000,5600,4400,3500,3000,2600,2300,210
B3
47.03,70.5,94.1,117.6,141.1,164.6,188.1,211.6,235.1,258.6,282.2,305.7,329.2,352.7,376.2,400,423.2,446.7,470.2
450000,280000,150000,80000,50000,35000,24000,18000,14000,9200,7000,5600,4400,3500,3000,2600,2300,210

4.2.5 File "buses.txt"

Vitovt Max Electro E433,E433
B1
Vitovt Electro E420,E420
B1
Model E321,E321
B2

4.2.6 File "routes.txt"

Railway Station - DS Viasnjanka,A1,9
Dolgobrodskaja - DS Siarova,T59,12.2
DS Drugnaja - DS Siarova,T43,7

4.2.7 File "trips.txt"

//Full name,short name,departure time,arrival time
Trip 1,T1,5.13,22.44
T43
Trip 2,T2,5.27,12.48
T43
Trip 3,T3,15.56,22.16
T43
Trip 4,T4,5.45,23.08
T43
Trip 5,T5,6,22.27
T43

4.2.8 File "fleet.txt"

2816,2816,E433,350
B1
T1
2818,2818,E433,700
B1,B2
T2,T3
2807,2807,E433,350
B1
T4
2805,2805,E433,350
B1
T5

4.2.9 File "febuses.txt"

2816
94.05
470.25
1
-376.2
2818
94.05
470.25
1
-160.74
94.05
470.25
1
-136.8
94.05
470.25
1
-160.74
94.05
470.25
1
-136.8
2807
94.05
470.25
1
-376.2
2805

94.05
470.25
1
-352.26

4.2.10 File "fpower.txt"

5,12,20,24
1,1,1,1
5,12,20,24
0.065,0.065,0.065,0.065
200,400,600,800
2000,4000,6000,8000

5 Output files for Minsk case

In this section output for data in previous section is provided in JSON and text formats.

5.1 File *solution.json* for Minsk case

{"UR": {
"pd": 200,
"c": "CS",
"K": 1,
"ttu": "16:33",
"stu": 0.69,
"cst": 31943.1,
"tu_i": ["0:07","0:18","0:15","6:00","0:48","2:17","4:04","2:16","0:11","0:17"],
"ts_i": ["0:00","5:27","5:45","6:00","12:00","12:48","15:56","20:00","22:16","22:27"],
"tf_i": ["5:00","5:45","6:00","12:00","12:48","15:56","20:00","22:16","22:27","22:44"]
}
, "CHB": [
{
"sn_eb": "2816",
"sn_bat": "B1",
"ct_eb": "2:00",
"SOC_d": [0],
"T_d": ["5:13"],
"SOC_a": [0],
"T_a": ["22:44"],
"ct_i": ["0:44","0:24","0:52"],
"ts_i": ["0:00","22:44","23:08"],
"tf_i": ["5:00","23:08","24:00"]
}
,
{
"sn_eb": "2818",
"sn_bat": "B1",
"ct_eb": "1:35",
"SOC_d": [0.28,0.23],
"T_d": ["5:27","15:56"],
"SOC_a": [0.22,0],
"T_a": ["12:48","22:16"],
"ct_i": ["0:17","0:13","0:14","0:51"],
"ts_i": ["0:00","5:00","5:13","12:48"],
"tf_i": ["5:00","5:13","5:27","15:56"]
}
,
{
"sn_eb": "2807",
"sn_bat": "B1",
"ct_eb": "2:00",
"SOC_d": [0],
"T_d": ["5:45"],
"SOC_a": [0],
"T_a": ["23:08"],
"ct_i": ["2:00"],

"ts_i": ["0:00"],
"tf_i": ["5:00"]
}
,
{
"sn_eb": "2805",
"sn_bat": "B1",
"ct_eb": "1:52",
"SOC_d": [2],
"T_d": ["6:00"],
"SOC_a": [0],
"T_a": ["22:27"],
"ct_i": ["1:52"],
"ts_i": ["0:00"],
"tf_i": ["5:00"]
}
]}

5.2 File *solution.out* for Minsk case

The optimal value of supplied power to the depot=200
Station <Charging station>
Number of stations=1
E-bus <2816>
Battery <Battery 1>
SOC at departure at 5:13 = 470.25 SOC at arrival at 22:44 = 94.05
Total charging time=2:00
Bus is charged:
0:44 in period [0:00,5:00]
0:24 in period [22:44,23:08]
0:52 in period [23:08,24:00]
E-bus <2818>
Battery <Battery 1>
SOC at departure at 5:27 = 470.25 SOC at arrival at 12:48 = 309.51
SOC at departure at 15:56 = 470.25 SOC at arrival at 22:16 = 333.45
Total charging time=1:35
Bus is charged:
0:17 in period [0:00,5:00]
0:13 in period [5:00,5:13]
0:14 in period [5:13,5:27]
0:51 in period [12:48,15:56]
E-bus <2807>
Battery <Battery 1>
SOC at departure at 5:45 = 470.25 SOC at arrival at 23:08 = 94.05
Total charging time=2:00
Bus is charged:
2:00 in period [0:00,5:00]
E-bus <2805>
Battery <Battery 1>
SOC at departure at 6:00 = 470.25 SOC at arrival at 22:27 = 117.99
Total charging time=1:52
Bus is charged:
1:52 in period [0:00,5:00]
Unused charging time resource 0:07 in period [0:00,5:00]
Unused charging time resource 0:18 in period [5:27,5:45]

Unused charging time resource 0:15 in period [5:45,6:00]
Unused charging time resource 6:00 in period [6:00,12:00]
Unused charging time resource 0:48 in period [12:00,12:48]
Unused charging time resource 2:17 in period [12:48,15:56]
Unused charging time resource 4:04 in period [15:56,20:00]
Unused charging time resource 2:16 in period [20:00,22:16]
Unused charging time resource 0:11 in period [22:16,22:27]
Unused charging time resource 0:17 in period [22:27,22:44]
Total unused daily charging time resource 16:33
The share of the used daily resource of charging time 0.69
The total cost=31943.1