PACE_2016_transit_feeds

— A PACE 2016 Challenge Instance Set (Track A: Treewidth) —

May 20, 2016 by J. Fichte, TU Wien

Short Description

The benchmark set PACE_2016_transit_feeds contains graph instances that have been extracted from publicly available real world mass transit data feeds [18] of cites, or metropolitan areas, or countries, e.g., Berlin metro region, DB long-distance traffic. We have split the graphs by transportation type, e.g., train, metro, tram, bus, and combinations thereof to obtain a wider picture. The transit feeds have been provided by various transit agencies according to the General Transit Feed Specification (GTFS). GTFS defines an interoperable format for public transportation schedules and associated geographic information and is a fundamental format for Google Maps Transit [12]. Unfortunately, many transit agencies do not provide the data under an open data license. Therefore, we prepared (i) a repository [6] that contains example graphs under CC-BY and present (ii) programs to download feeds from public APIs (get_feeds.py) and to extract data from a GTFS transit feed (gtfs2graphs.py) [5].

Graph format

Instances in the set PACE_2016_transit_feeds are encoded in the DIMACS-like graph format "gr". For a general description of the graph format "gr" we refer to Appendix A of the PACE 2016 descriptions [3]. Note that we extended this format by storing the name of the original transit file, the symbol table (for vertex names), vertex labels (latitude and longitude), and edge labels as comments on the head of the instance file.

```
c Contains a public transit graph extracted from GTFS file
c original_input_file: somegtfs.zip
c ----------------------------------------
c Symbol Table
c tab 1 |
c tab 2 | 23
c ----------------------------------------
c Edge Labels
c edge (1, 2) | {'route_type': 200, 'agency': 'TransLink', 'weight': 120, 'area': 'BC'}
c ----------------------------------------
c Vertex Labels
c ----------------------------------------
p tw 2 2
1 2
```

Public Transit Feeds

The example graph instances [6] are available at https://github.com/daajoe/PACE2016_transit_graphs. The repository currently contains transit graphs for: (i) the metropolitan region of Berlin and the state of Brandenburg [16] (the GTFS transit feed was provided by the open data portal of the government of Berlin), (ii) DB Long-distance Traffic for Western Europe [9] (the GTFS transit feed has been extracted using a tool by Brosi [2] from the open data API of the German railroad company and cleaned from invalid data fields).

Transit Feed Converter (gtfs2graphs.py)

The program gtfs2graphs.py extracts graphs from a given GTFS transit feed file [5] into various graph formats (gml [10], lp [17], dimacs [15], gr [3]). The source code is available at https://github.com/daajoe/gtfs2graphs. The tool is written in Python (version 2.7) and licensed under GPL2 [8]. To run the tool, please download the most recent source code and consult the included file README.md. Make sure that the target platform has the required software installed. In particular, a recent version of the python library transitfeed [13] needs to be installed. Since transitfeed is by default very verbose, we provide a fork [7] where verbosity of the output can be modified.

✉Johannes.Fichte@tuwien.ac.at

After downloading `gtfs2graphs.py`, one can check whether everything works fine by invoking the program with the flag "`--help`", which provides the available command line options.

You can generate a graph in the format "gr" of the feed "somegtfs.zip" by running `./gtfs2graphs.pyfeeds/somegtfs.zip` in the directory "`gtfs2graphs/gtfs2graphs`". The output graph will then be stored by default in "`feeds/gr/somegtfs.zip.gr`". An alternative path of the output file can be specified by means of the flag "`--output_file=path`". If you want to output the graph to stdout instead of a file, add the flag "`--stdout`". If you want to run `gtfs2graphs.py` on all feeds in the directory "`feeds`" and check whether is produced a valid gr-file (using the program "`td-validate`" that is stored in the directory "`~/src/td-validate`", you can simply run "`./run4allfeeds.sh -c ~/src/td-validate feeds`" in the directory "`gtfs2graphs/gtfs2graphs`".

By default the program also extracts graphs depending on route types, e.g., bus, train, metro, tram, and combinations thereof, e.g., "`feeds/gr/somegtfs.zip_bus.gr`", "`feeds/gr/somegtfs.zip_tram.gr`", "`feeds/gr/somegtfs.zip_tram+bus.gr`". You can modify the splitting behavior by editing lines in section "types" of the configuration file "`conf/extract_route_types_conf.yaml`". For example the line `'bus':[3,200..300,700..800]` instructs the program to create a graph where the route types are among 3, between 200 and 300, and so forth. For detailed information about values of GTFS route types we refer to the files "`conf/route_types.csv`" and "`conf/route_types_extended.csv`" and other sources [11, 14]. Splitting can be disabled by adding the flag "`--no_split`".

If you want to specify a mapping for a transport agency to areas modify the mapping file "`conf/gtfs_info_conf.yaml`".

GTFS downloader (`get_feeds.py`)

Many transit agencies around the world make their schedules available as GTFS transit feeds, see e.g., GTFS data exchange [4] or TransitFeeds [18]. However, the data is often provided under various (limited sharing) data licenses. Therefore, we currently give only certain example instances, which have been licensed under CC-BY. However, our program `get_feeds.py` downloads all GTFS transit feeds available at `http://transitfeeds.com/feeds` [18]. The source code is available at `https://github.com/daajoe/gtfs2graphs`. The tool is written in Python (version 2.7) and licensed under GPLv2 [8]. To run the tool, please download the most recent source code and consult the included file README.md. Make sure that the target platform has the required software installed. After downloading `get_feeds.py`, one can check whether everything works fine by invoking the program with the flag "`--help`", which provides the available command line options.

In order to run the program, you need an API key from `http://transitfeeds.com/api/keys`. After you downloaded your API key, you need to place the key in file: "`conf/get_feeds_conf.yaml`" behind `key:` in quotation marks, e.g., `key:'some-api-key'`. You can start the download of transit feeds by running `get_feeds.py`, which will start the download of the available feeds into the folder "`gtfs2graphs/feeds/`". We suggest to use symlinks to modify the location of the folder. Be aware that a full download may require several gigabytes. You can abort a download by pressing `Ctrl+C`. To abort the program hit `Ctrl+\`.

Note that we checked all downloaded feeds at the time of the submission using the checker `td-validate` [1].

Acknowledgements

The author would like to thank Andreas Pfandler for helpful comments on the draft and for beta-testing the programs.

References

[1] Cornelius Brand et al. "holgerdell/td-validate: Tree decomposition validity checker". `https://github.com/holgerdell/td-validate`. 2016.

[2] Patrick Brosi. "patrickbr/db-api-to-gtfs: db-to-gtfs". `https://github.com/patrickbr/db-api-to-gtfs`. 2015.

[3] Holger Dell. "The 1st Parameterized Algorithms and Computational Experiments Challenge – Track A: Treewidth". https://pacechallenge.wordpress.com/track-a-treewidth/. 2016.

[4] Jehiah Czebotar et.al. *GTFS data exchange*. http://www.gtfs-data-exchange.com.

[5] Johannes K. Fichte. "daajoe/gtfs2graphs – A GTFS transit feed to Graph Format Converter". https://github.com/daajoe/gtfs2graphs. 2016.

[6] Johannes K. Fichte. "daajoe/PACE2016_transit_graphs – Transit feeds A PACE 2016 Competition Instance Set". https://github.com/daajoe/PACE2016_transit_graphs. 2016.

[7] Johannes K. Fichte. "daajoe/transitfeed – Fork of the google GTFS transit feed library transitfeed". https://github.com/daajoe/transitfeed. 2016.

[8] The Free Software Foundation. "GNU General Public License". http://www.gnu.org/licenses/gpl.html. 2007.

[9] DB Vertrieb GmbH. "DB Open Data – Fahrplan". http://data.deutschebahn.com/apis/fahrplan/. 2016.

[10] Michael Himsolt. "GML: A portable Graph File Format". http://www.fim.uni-passau.de/index.php?id=17297&L=1. Passau, Germany, 2009.

[11] Google Inc. "General Transit Feed Specification Reference – routes.txt". https://developers.google.com/transit/gtfs/reference#routestxt. 2016.

[12] Google Inc. "Google Transit FAQs". http://maps.google.de/help/maps/mapcontent/transit/faq.html. 2016.

[13] Google Inc. "google/transitfeed – A Python library for reading, validating, and writing transit schedule information in the GTFS format." https://github.com/google/transitfeed. 2015.

[14] Google Inc. "Transit Partners Help – Extended GTFS Route Types". https://support.google.com/transitpartners/answer/3520902?hl=en. 2016.

[15] Bart Massey. "Coloring Problems DIMACS Graph Format". http://prolland.free.fr/works/research/dsat/dimacs.html. 2001.

[16] Verkehrsverbund Berlin-Brandenburg (VBB). "VBB-Fahrplandaten Ende April bis Dezember 2016". http://daten.berlin.de/datensaetze/vbb-fahrplandaten-ende-april-bis-dezember-2016. 2016.

[17] Johan Wittocx. *ASP Competition 2013 – Problem Description: Maximal Clique Problem*. https://www.mat.unical.it/aspcomp2013/MaximalClique. 2013.

[18] Quentin Zervaas and et al. *GTFS data exchange*. http://transitfeeds.com. 2016.