Steinlib Documentation

Release 0.1

Leandro Nunes

Apr 28, 2017

Contents:

This module provides a Python parser to STP, the SteinLib Testdata Library file format.

The SteinLib Testdata consists of hundreds of Steiner tree problem instances. Using this module allows you to write your own Python code to handle STP format parsed and validated structures.

There is complete documentation about the format itself on the official steinlib website.

CHAPTER 1

Installing

It is easy to obtain the latest released version via `pip`:

```
pip install steinlib
```

CHAPTER 2

Basic Usage

In order to use the parsed structures, two main components are required:

- A *parser*, that will be responsible for parsing the structure, and;
- A *instance*, where the parser will invoke callbacks when known structures are found

So, let's see an example about how it works.

Starting with an STP file available on Steinlib official website:

```
33D32945 STP File, STP Format Version 1.0

SECTION Comment
Name    "Odd Wheel"
Creator "T. Koch, A. Martin and S. Voss"
Remark  "Example used to describe the STP data format"
END

SECTION Graph
Nodes 7
Edges 9
E 1 2 1
E 1 4 1
E 1 6 1
E 2 3 1
E 3 4 1
E 4 5 1
E 5 6 1
E 6 7 1
E 7 2 1
END

SECTION Terminals
Terminals 4
T 1
T 3
```

```
T 5
T 7
END

SECTION Coordinates
DD 1  80 50
DD 2  30 50
DD 3  55  5
DD 4 105  5
DD 5 130 50
DD 6 105 95
DD 7  55 95
END

EOF
```

It is possible to observe this file contains three main components:

- a **header line**;
- a set of delimited **sections** and;
- the **end of file** marker.

This *parser* will identify all these small pieces of the STP file and use it to trigger functions on your *instance*. This is the key concept used to create this module.

The script below provides some initial examples about how these `callback` methods works:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from steinlib.instance import SteinlibInstance
from steinlib.parser import SteinlibParser

import sys


class MySteinlibInstance(SteinlibInstance):
    """
    This is my first steinlib parser!
    """

    def comment(self, raw_args, list_args):
        print "Comment section found"

    def comment__end(self, raw_args, list_args):
        print "Comment section end"

    def coordinates(self, raw_args, list_args):
        print "Coordinates section found"

    def eof(self, raw_args, list_args):
        print "End of file found"

    def graph(self, raw_args, list_args):
        print "Graph section found"

    def header(self, raw_args, list_args):
        print "Header found"

```

```
32      def terminals(self, raw_args, list_args):
33          print "Terminals section found"
34
35
36  if __name__ == "__main__":
37      my_class = MySteinlibInstance()
38      with open(sys.argv[1]) as my_file:
39          my_parser = SteinlibParser(my_file, my_class)
40          my_parser.parse()
41
```

For each identified structure, the appropriate method will be called in the *instance* object, with the standard parameters:

- `raw_args` is the actual full line from the input stream
- `list_args` contains the extracted and converted parameters from the current line