1 MOP File format

The PolySCIP file format (with suffix `.mop`) is based on the widely used `MPS` file format (see [1], [2]). `MPS` is column-oriented and all model components (variables, rows, etc.) receive a name. An objective in `MPS` is indicated by an `N` followed by the name in the `ROWS` section. Similarly, in the `MOP` format the objectives are indicated by `N` followed by the name in the `ROWS` section. In general, `MPS` might not be as human readable as other formats. However, one of the main reasons to base the file format of PolySCIP on it is its easy extension towards several objectives and its wide availability in most of the linear and integer programming software packages such that available `MPS` parsers could easily be adjusted to parse an `.mop` file as well. Furthermore, no user is expected to write `.mop` files by hand, but to use a modelling language that does the job. See Section 2 for a description of how to use the freely available Zimpl and the Python script `mult_zimpl_to_mop.py` (comes with PolySCIP) to generate `.mop` files.

The following simple equation-based bi-criteria integer problem

$$
\begin{array}{lll}
\text{maximize} & \text{Obj1:} & 3x_1 + 2x_2 - 4x_3 \\
& \text{Obj2:} & x_1 + x_2 + 2x_3 \\
\text{subject to} & & \\
& \text{Eqn:} & x_1 + x_2 + x_3 \quad = 2 \\
& \text{Lower:} & x_1 + 0.4x_2 \quad \leq 1.5 \\
& & x_1,\ x_2,\ x_3 \quad \geq 0 \\
& & x_1,\ x_2,\ x_3 \quad \in \mathbb{Z}
\end{array}
$$

is written in MOP format as follows:

```
NAME          BICRIT
OBJSENSE
 MAX
ROWS
 N  Obj1
 N  Obj2
 E  Eqn
 L  Lower
COLUMNS
    x#1       Lower             1
    x#1       Eqn               1
    x#1       Obj2              1
    x#1       Obj1              3
    x#2       Lower           0.4
    x#2       Eqn               1
    x#2       Obj2              1
    x#2       Obj1              2
```

```
    x#3       Eqn              1
    x#3       Obj2             2
    x#3       Obj1            -4
RHS
    RHS       Eqn              2
    RHS       Lower          1.5
BOUNDS
 LI BOUND     x#1              0
 LI BOUND     x#2              0
 LI BOUND     x#3              0
ENDATA
```

2 User-friendly `.mop` file generation

Zimpl is a freely available modelling language (also part of the SCIP Optimization Suite) to translate a mathematical model of a problem into a mathematical program in `.mps` (or `.lp`) file format. Together with the `mult_zimpl_to_mop.py` script (located in the 'mult_zimpl' directory of PolySCIP) it can/should be used to generate your `.mop` files. For a more detailed description of Zimpl, see the Zimpl user guide. In this section we will only describe how to make use of it, but not all options how to write different models. Zimpl does generally not support several objectives; this is where `mult_zimpl_to_mop.py` comes into play. It takes an 'extended' Zimpl file containing several objectives, internally rewrites all but the first objectives into constraints, calls Zimpl on the rewritten file and changes the file generated by Zimpl containing 'artificial' constraint indicators back to objective indicators which yields an `.mop` file.

- Zimpl comes with the SCIP Optimization Suite
  - Please see the INSTALL file of the SCIP Optimization Suite (you basically just need the GMP library in order to build).
- `mult_zimpl_to_mop.py` is a Python3 script and comes with PolySCIP; it is located in the 'mult_zimpl' directory
  - Execute `python3 mult_zimpl_to_mop.py your_model.zpl` to run it on the file `your_model.zpl` containing your multi-criteria model
  - The following command line arguments are available

    -h, --help Shows the help message and exits

    -p <path> The directory where the generated `.mop` file should be saved

    –path_to_zimpl <path> The directory where your *zimpl* binary is located

  E.g., if Zimpl was not installed globally but in `/home/user/zimpl`, and, furthermore, if you would like to save the generated `.mop` file in `/tmp/`, then run `python3 mult_zimpl_to_mop.py -p /tmp/ -path_to_zimpl /home/user/zimpl/bin your_model.zpl`

Please note (in the following examples) that the direction of optimization, i.e., `minimize` or `maximize`, is declared only once followed by the first objective. All other objectives follow without a direction specification implying that all objectives are assumed to be either minimized or maximized.

Example 2.1. The bi-criteria maximization problem of Section 1 could be modelled as follows:

```
set I := {1..3};
param c1[I] := <1> 3, <2> 2, <3> -4;     #coefficients of the first objective
param c2[I] := <3> 2 default 1;          #coefficients of the second objective
param low[I] := <1> 1, <2> 0.4, <3> 0;   #coefficients of the lower constraint
var x[I] integer >= 0;

maximize Obj1: sum <i> in I: c1[i]*x[i];
Obj2: sum <i> in I: c2[i]*x[i];

subto Eqn: sum <i> in I: x[i] == 2;
subto Lower: sum <i> in I: low[i]*x[i] <= 1.5;
```

Saving the file, e.g., as test.zpl and running `mult_zimpl_to_mop.py` on it would generate a file named `test.mop` which can be solved with PolySCIP.

Example 2.2. A tri-criteria linear programming minimization problem with a four-dimensional standard cube as feasible space and three unit vectors as objectives could be modelled as follows:

```
set I := {1..4};
var x[I] real;
minimize Obj1: x[1];
Obj2: x[2];
Obj3: x[3];
subto Bounds: forall <i> in I: 0 <= x[i] <= 1;
```

Again, saving the file, e.g., as testCube.zpl and running `mult_zimpl_to_mop.py` on it would generate a file named `testCube.mop` which can be solved with PolySCIP.

For more involved models there is also the possibility to read external files containing data for parameters. Please the see the Zimpl user guide for more details.

References

[1] MPS format (short), https://en.wikipedia.org/wiki/MPS_%28format%29

[2] MPS format (detailed), http://lpsolve.sourceforge.net/5.5/mps-format.htm