Introduction to PySCIPOpt

March 6, 2018

PySCIPOpt is the Python interface for SCIP. It allows for *fast model prototyping* with *flexible expressions*. Additionally PySCIPOpt supports *user-plugins* similar to C.

```
from pyscipopt import Model

# initialize model
m = Model("svm")

# add variables
x = m.addVar(vtype='B', name='x')
y = m.addVar(vtype='C', name='y')

# add a constraint
m.addCons(3 * x + 2 * y >= 4)

m.optimize()
```

Adding variables

Different variable types `vtype` are supported by `addVars`: `B` (binary), `I` (integer), `C` (continuous). You can specify lower and upper bounds: `lb` (default: 0), `ub` (default: `None` $\sim \infty$), as well as objective coefficients: `obj` (default: 0.0).

```
m = Model("svm")

# add variables
x = m.addVar(vtype='C', name='x', lb=-10, ub=10, obj = 1.0)
```

Nonlinear objective functions

SCIP can only handle linear objective functions. Therefore a nonlinear objective function must be transformed into a constraint using an auxiliary variable:

$$\min f(x) \Leftrightarrow \min t \text{ such that } f(x) \leq t$$

Adding constraints

`addCons` understands expression objects and requires an operator `<=`, `>=`, `==`. Variables and constants can appear on both sides of the operator. For complex expressions use the `quicksum` summation and python list comprehension `[a[i] for i in range(5)]`, multiplying variables yields nonlinear constraints,

```
from pyscipopt import Model, quicksum
m = Model("svm")
vars = []
for i in range(10):
    vars.append(m.addVar(vtype='B', name=str(i)))
m.addCons(vars[0] ** 2 == 1)
m.addCons(quicksum(i * vars[i] for i in range(10) <= 15),
          name="mycons")

m.optimize()
```