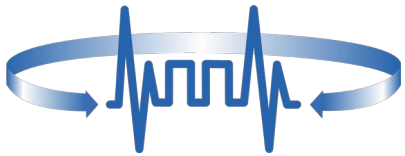
The 11th International Conference "Distributed Computing and Grid Technologies in Science and Education" (GRID'2025)

Adaptation of parallel implementations of the SCIP global and discrete optimization solver to AMPL input and output data formats

Sergey Smirnov, Vladimir Voloshinov

Center for Distributed Computing,
Institute for Information Transmission Problems
of the Russian Academy of Sciences (Kharkevich Institute)



The research was carried out within the state assignment of Ministry of Science and Higher Education of the Russian Federation for IITP RAS

Outline

- What is SCIP, FiberSCIP & ParaSCIP?

  [Gurobi, CPLEX, Xpress, OptVerse, **SCIP,** HiGHS, CBC]

- Input/Output formats in SCIP
- Why AMPL, ampl.com, and Pyomo, www.pyomo.org?
- There is a SCIP's interface in Pyomo
- Problem: no interface FiberSCIP&ParaSCIP with Pyomo
- Our approaches to solve the problem

What problems SCIP solves

Mathematical Programming &
Constrained Programming Problems:

Feasible domain of mixed integer variables

$$Q \doteq \{(x_c, x_z) \in \mathbb{R}^{n_c} \times \mathbb{Z}^{n_z} : f_i(x_c, x_z) \leqslant 0 (i \in \mathbf{I}), g_j(x_c, x_z) = 0 (j \in \mathbf{J})\}$$

Mathematical Programming (Mixed Integer NLP)

$$f_o(x) \to \min,\ x \in Q, x = (x_c, x_z), x_c \in \mathbb{R}^{n_c}, x_z \in \mathbb{Z}^{n_z}$$

Constraint Programming (Mixed Integer ones)

$$\text{find some } x^* \in Q \text{ or prove that } Q = \varnothing$$

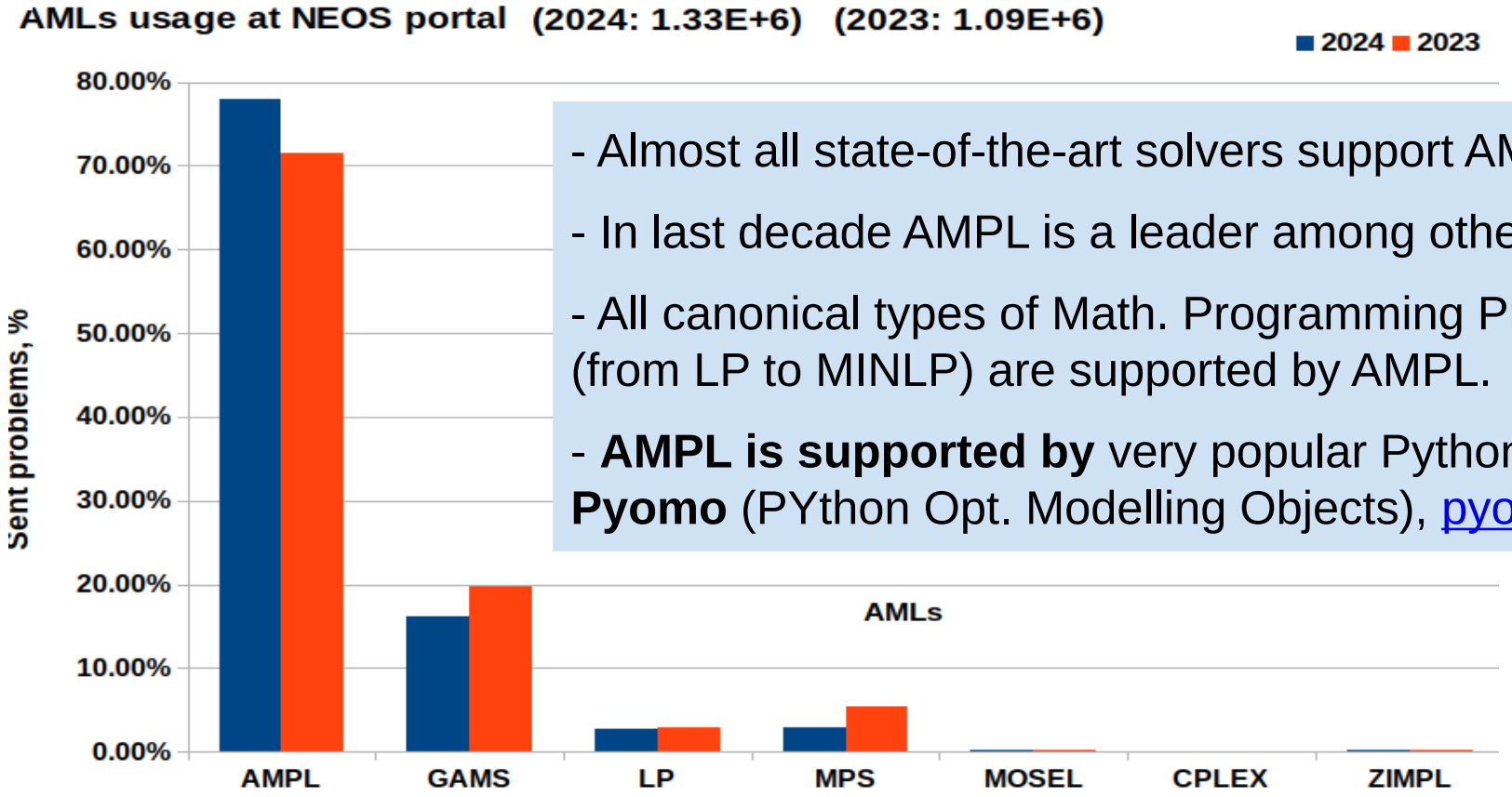
SCIP stands for **Solving Constraint Integer Programs**

SCIP solver

- A very modular CIP/MINLP/Global opt. solver framework
- Not as fast as Gurobi/CPLEX but rather universal (MINLP !)
- Free and open-source (Apache license, since Nov. 2022)
- https://www.scipopt.org, ***Zuse Institute Berlin (ZIB)***
- ***scip*** (executable, one threaded mainly)
- Parallel implementations (***fscip***, ***parascip***):
  - FiberSCIP = UG[SCIP, pthreads] – shared memory
  - ParaSCIP = UG[SCIP, MPI] – distributed memory
  - *Concurrent mode in SCIP itself* (not as effective)

Why AMPL (since ~1987)?

Rating popularity of AMLs (Algebraic Modelling Languages):
“format” of opt. problems **sent to various solvers** on NEOS-server.org.



- Almost all state-of-the-art solvers support AMPL.
- In last decade AMPL is a leader among other AMLs.
- All canonical types of Math. Programming Problems (from LP to MINLP) are supported by AMPL.
- **AMPL is supported by** very popular Python package **Pyomo** (PYthon Opt. Modelling Objects), pyomo.org

SCIP's Input and Output Formats

- Input model formats: NL-file (with -AMPL flag), MPS, LP (CPLEX), CIP (SCIP's native) etc.
  (https://www.scipopt.org/doc-9.2.2/html/group__FILEREADERS.php)
- Output solution formats: .sol (with -AMPL flag) or **.cip**
- AMPL output format is supported by *scip* executable only

  MINLP->NL(AMPL)->*scip*->SOL(AMPL)->analysis
- FiberSCIP and ParaSCIP support all the same input model formats but only raw SCIP (.cip) output format

  MINLP->NL(AMPL)->*fscip*->CIP(SCIP)**->???**

MILP / MINLP Optimization in Python

- **Pyomo** – **MINLP for many solvers + AMPL support**
- PuLP – MILP for many solvers
- Python-MIP – MILP, deep integration with CBC and Gurobi
- SciPy – MILP with HiGHS solver
- Solver-specific Python packages (via Python APIs):
  - Gurobipy (Gurobi)
  - PySCIPopt (SCIP)
  - Highspy (HiGHS solver)
  - etc.

Pyomo (PYthon Optimization Modelling Objects)

- Python-based open-source optimization modeling package
- Pyomo Algebraic Modeling Language (AML)
- Supports all canonical Mathematical Programming Problem types including MINLP
- Many solver interfaces (free and commercial ones)
- For SCIP only AMPL, ampl.com, interface is supported
- **No FiberSCIP & ParaSCIP support**

Our problem

- How to use parallel SCIP from Pyomo?
- Pyomo has only AMPL interface to SCIP (SCIPAMPL.py): writes model as NL file, expects solution in AMPL SOL format: *Pyomo->NL(AMPL)->scip->SOL(AMPL)->Pyomo*
- **Recently (!) last versions** Fiber/ParaSCIP can read NL files, but write solutions in CIP-format only still.
- How to solve mixed-integer non-linear programs (MINLP) with FiberSCIP or ParaSCIP by Pyomo?

Possible Approaches

- Direct approach:
  - Implement FiberSCIP Python interface for Pyomo (e.g. PuLP has one for FiberSCIP, but for LP/MILP only!)
- Workaround:
  - To write a wrapper Bash-script for ***fscip*** executable to mimic it as an AMPL-compatible SCIP solver for Pyomo.
  - It converts NL-input (problem & initial solution if any), then uses solution in CIP-format and ***fscip*** solver's logs to write AMPL SOL.

Our approach (1)

How ***SCIPAMPL.py*** Pyomo plugin works:

- During solver discovery it requests solver's version
- Prepares a file with solver's options in a temporary directory
- After solution completes, it reads solution and logs

	scip	**fscip**
Get version details	$ scip --version	N/A
Solve	$ scip problem -AMPL	$ fscip ug.set problem.nl
Solution values	Read from .sol	Read from **CIP**.sol and **convert to AMPL SOL**
Solution status	Read from .sol (solution found, errors)	Read from stdout
Primal/dual bounds, solving time	Read from stdout (primal value, gap value)	Read from stdout

Our approach (2)

We've implemented a wrapper bash script

https://github.com/distcomp/SvF/blob/docker/ugscip/run-fscip-pyomo.sh

NL(AMPL)->*fscip*->sol(CIP)->**scip**->sol(AMPL)

Main steps:

- Execute ***scip --version*** instead of ***fscip*** if executed with --version
- Call FiberSCIP
- Convert solution
- Parse solver logs to get solution status (absent from *fscip*'s sol(CIP))

Just the same may be done for ParaSCIP (e.g. on cluster or on the same host with multiple CPUs)!

Our approach (3)

We use capabilities of SCIP application console commands !

SCIP may be used as format converter by SCIP **console commands**:

- for $NAME.nl (AMPL) -> $NAME.cip, *.cip.init.sol (CIP)

  $ echo "**write problem** $NAME.cip **write solution** $NAME.cip.init.sol **quit**" | **scip** $NAME -AMPL **-i**

  *It is obsolete due to that now **fscip** can read NL directly!*

- for $NAME.cip, $NAME.cip.sol -> $NAME.sol (AMPL)

  $ echo "**read** $NAME.cip.sol **quit**" | **scip** $NAME -AMPL **-i**

  See https://github.com/distcomp/SvF/blob/docker/ugscip/run-fscip-pyomo.sh

Run FiberSCIP from Pyomo

- Example of Python-code (similar to calling other solvers)

```
import pyomo.environ as pyo

model = pyo.ConcreteModel()
model.x = pyo.Var([1,2], domain=pyo.NonNegativeReals)
model.OBJ = pyo.Objective(expr = 2*model.x[1] + 3*model.x[2])
model.Constraint1 = pyo.Constraint(expr = 3*model.x[1] + 4*model.x[2] >= 1)

opt = pyo.SolverFactory('scip', executable='./run-fscip-pyomo.sh')
opt.solve(model)

print(pyo.value(model.x[1]))
print(pyo.value(model.x[2]))
```

```
$ python3 test.py
0.3333333333343035
0.0
```

Conclusion

- Fiber/ParaSCIP is a high performance shared-memory parallel version of SCIP.
- We have Implemented a wrapper Bash-script, which allows to use FiberSCIP solver directly from Pyomo.
- Similar Bash-script may be used to use ParaSCIP.
- The solution have a good balance of complexity and ease of use for experimentation.

Thank you!

Vladimir Voloshinov, Sergey Smirnov
IITP RAS (Kharkevich Institute)
vladimir.voloshinov@gmail.com, sasmir@gmail.com

Future plans

Simplify the solution by skipping the model conversion: NL(AMPL)->*fscip*->sol(CIP)->scip->sol(AMPL) - need to check if initial solution will be handled properly

Maybe implement an FSCIP plugin for Pyomo