Introduction to ASOL – Affine Solver

Description

The aim of the implemented branch-and-prune algorithm is to find all solutions to systems of nonlinear equations. The number of variables must equal the number of equations ("square systems"). The variables must have proper lower and upper bounds ("box constrained"). The problem is supposed to be given in the AMPL modeling language. ASOL is a proof-of-concept implementation, have realistic expectations. Download the executable binary and the source code from its website.

Mathematical background

The branch-and-prune algorithm is presented at a conceptual level in the manuscript entitled Computing multiple steady states in homogeneous azeotropic and ideal two-product distillation. Do not let the title scare you away, the part presenting the algorithm is human readable. Outline of the solver modules are given on the next page.

Using ASOL through AMPL

The AMPL environment was designed to deal with a single solution only. As a consequence, it is a bit uncomfortable to deal with several solutions. If you do not have AMPL, download the free student version for your platform. You will find supplementary material on the website. You can get help on the AMPL mailing list.
Write your model in the AMPL modeling language or simply take the eco9.mod model from the benchmarks directory. Once you are ready with your model, the first step is to create the .nl file. Assuming that ampl is on the PATH, enter the following in the command line:

```
ampl -ogeco9 eco9.mod
```

A new file eco9.nl should have appeared. Pass this to ASOL together with the -AMPL flag:

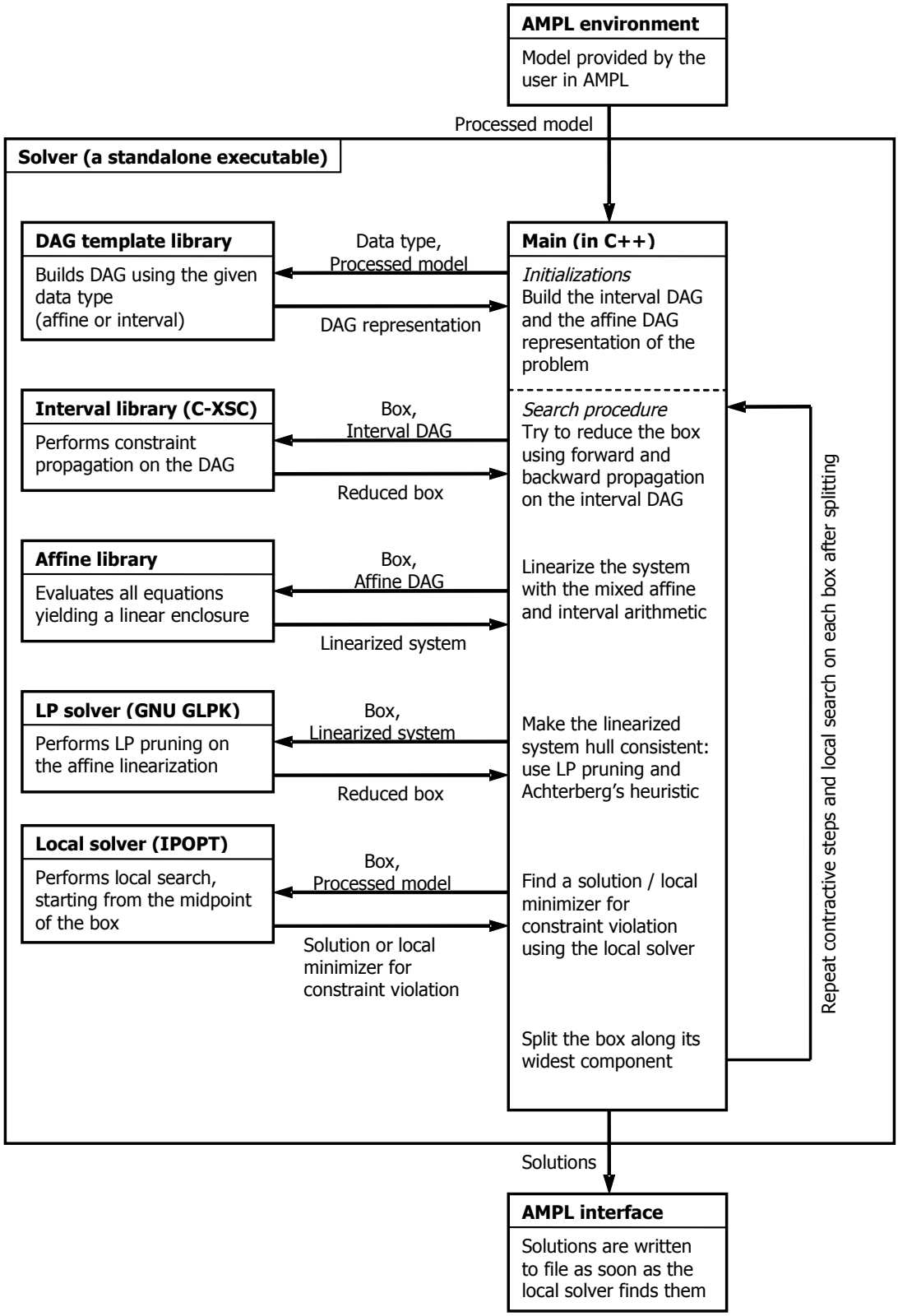
```
asol eco9.nl -AMPL
```

ASOL will print a lot of debug info, just ignore them. The found solutions will be written to files with .sol extension. You can analyze them in ampl as follows:

```
ampl
(starts the ampl shell)
ampl: model eco9.mod;
ampl: solution eco9_2.sol;
ampl: display x1, x2, x3;
x1 = 0.645506
x2 = 0.617128
x3 = 0.672842
```

Outline of the solver modules

Third party libraries are C-XSC, GNU GLPK, IPOPT, ASL (AMPL interface). See also Mathematical background.



Limitations of the current implementation

These limitations do NOT apply to the branch-and-prune algorithm itself. These are implementation flaws and they can be remedied, at least to a certain extent.

- Only a handful of functions are implemented: basic arithmetic operations (+, −, ·, /) and the following functions: ln, exp, sqr. Workarounds like replacing x^3 with $x^2 \cdot x$ may work but can significantly degrade performance. Also, keep in mind that AMPL can reformulate your model when generating the .nl file.
- You have to provide proper lower and upper bounds on the variables. Workarounds like giving big M variable bounds (such as 1.0E+8) not only degrades performance but can make the solver crash due to numerical issues.
- Certain problems can have continuous solution set(s). The solver will try to chop this continuous set(s) into tiny parts and return all of them. After all, these parts are solutions. The solver can return literally millions of solutions, leading to obvious issues.
- If the domain of a variable is reduced to a zero-width interval during any iteration, the solver stops with an error message. It is just an implementation flaw.
- ASOL was meant to be used with large and sparse systems. It shows relatively poor performance on tiny problems. Nevertheless, it can solve them in reasonable time.

The algorithm itself is perfect in the sense that it has to find all solutions to the problem. If it misses a solution it is due to the shortcomings of the implementation. Again, these issues can be resolved.

- First of all, the code is definitely not bug free.
- For various reasons, tiny coefficients have to be removed from the internally computed LP problems. This can introduce much larger error into the result than the rounding errors. As a consequence, the solver may erroneously cut out solutions.
- Rounding errors are ignored.
- If the domain of a variable is reduced to a narrow interval (or given as a narrow interval by the user), the affine linearization can become erroneous due to rounding errors.
- Huge variable bounds can result in overflow / underflow, ill-conditioned internal problems, *etc*. These numerical problems can result in losing solutions.
- Only high quality solutions are accepted. Probably "Solved To Acceptable Level" quality solutions should not be rejected. (Check the IPOPT documentation for further details.)
- When checking the solutions returned by the local solver IPOPT, both absolute and relative differences should be checked between the newly found and the stored solution vectors. Now, only the absolute differences are checked. This can result in losing solutions and spurious solutions.