## Supplementary Material

*DistributedFBA.jl* is part of *COBRA.jl* (see Fig. 1). The COBRA module wraps *load.jl*, *distributedFBA.jl*, and *solve.jl*. The input to the COBRA module is a *.mat* file that contains data of a COBRA model as defined in (Schellenberger *et al.*, 2011). This HDF5 model is loaded using the *MAT.jl* module (Kornblith *et al.*, 2012). Additionally, solver configuration parameters that are set in *solverCfg.jl*. are input to the COBRA module.

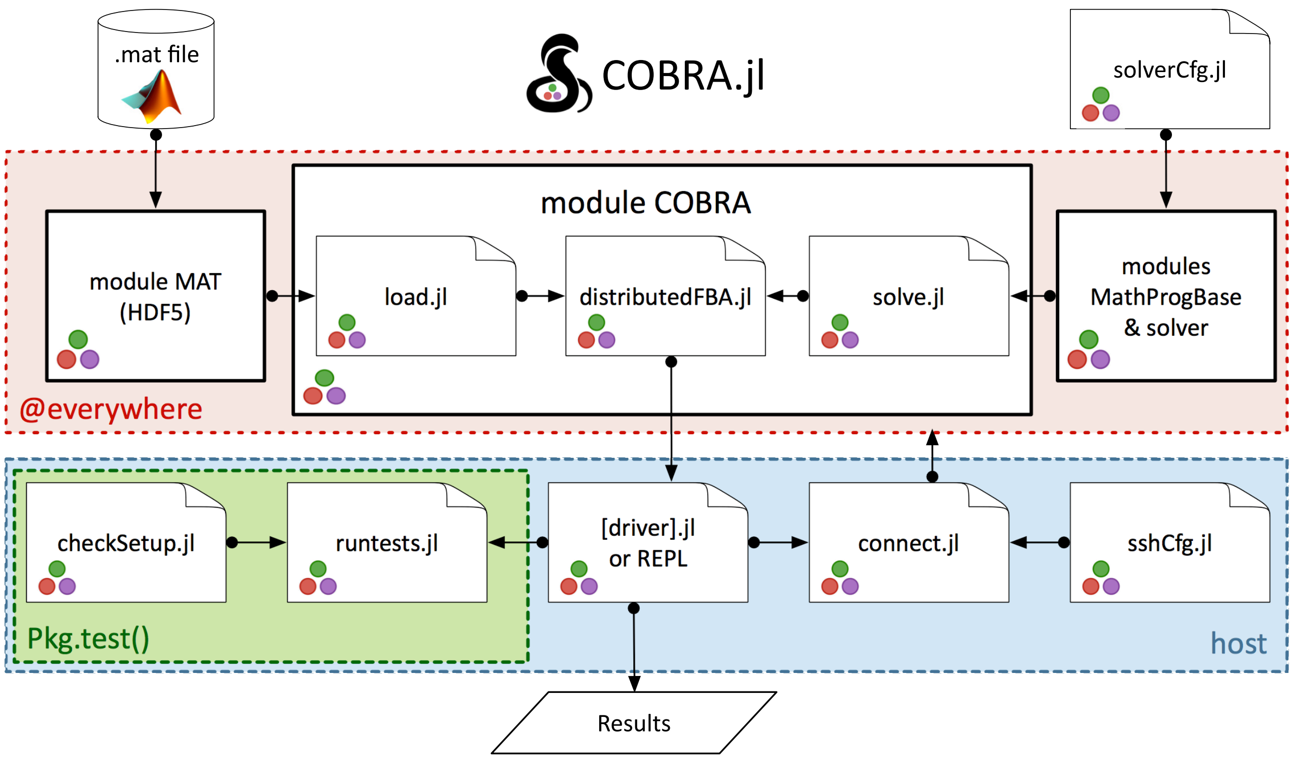

**Fig. 1.** Overview of the COBRA.jl package.

A parallel pool with either local or remote workers (using *connect.jl*) may be created using either the Julia REPL or a driver. The COBRA module and its dependencies, such as *MathProgBase.jl* (Lubin *et al.*, 2015) and solver interfaces, are spawned from the host node to each worker with the macro *@everywhere*. This ensures that the full model and the solver interfaces are available on each worker (including the host), although only a subset of the FBA problems are solved on each worker. The results are assembled on the host and fetched from the workers independent of the size of the parallel pool.

The core functions for distributing and solving multiple FBA problems are defined in the COBRA module. The main function within the COBRA module is *distributedFBA()* defined in *distributedFBA.jl*, which loads the model from file (*load.jl*: *loadModel()*), builds the LP model (*solve.jl: buildCobraLP()*), and maximises or minimises the LP problem (*solve.jl: solveCobraLP()*) on the spawned processes with a different set of FBA problems using *distributedFBA.jl: loopFBA()*. Before the LP problems are solved, additional constraints may be added to the model using *distributedFBA.jl: preFBA!()*. The FBA problems are distributed using *distributedFBA.jl: splitRange()* according to the splitting strategy $s$, which is based on the sorted column density vector $\rho_c$ of the stoichiometric matrix $S$:

- $s = 0$ : *Blind splitting*: default random distribution
- $s = 1$ : *Extremal dense-and-sparse splitting*: every thread receives dense and sparse reactions, starting from *extremal* indices of $\rho_c$
- $s = 2$ : *Central dense-and-sparse splitting:* every thread receives dense and sparse reactions, starting from the *central* indices of $\rho_c$

The COBRA module may be tested using *runtests.jl*, which also checks the computing node configuration (*checkSetup.jl*), and confirms that a compatible and working solver installation is present.

## References

Kornblith, S. *et al.* (2012) Support for reading and writing MATLAB files in Julia, *GitHub code*.
Lubin, M. *et al.* (2015) Computing in Operations Research using Julia, *INFORMS Journal on Computing*, **27**(2), 238–248, doi:10.1287/ijoc.2014.0623.
Schellenberger, J. *et al.* (2011) Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox v2.0, *Nature protocols*, **6**, 1290–1307.