Cairo university
Faculty of computers and information
Decision support department

Name: Ahmed Ehab Hussein
Id: 20160007
Group: CS_DS_1

Topic: JuMP

JuMP

is a domain-specific modeling language for mathematical optimization embedded in Julia.

Problems classes:

It currently supports a number of open-source and commercial solvers for a variety of problem classes, including:
1- linear programming
2- mixed-integer programming
3- second-order conic programming
4- semidefinite programming
5- and nonlinear programming.

JuMP's features include:

1- User friendliness
2- Speed
3- Solver independence
4- Access to advanced algorithmic techniques
5- Ease of embedding

User friendliness:

- Syntax that mimics natural mathematical expressions.
- Complete documentation (work in progress (WIP))

Speed:

- Benchmarking has shown that JuMP can create problems at similar speeds to special-purpose modeling languages such as AMPL.
- JuMP communicates with most solvers in memory, avoiding the need to write intermediary files.

Solver independence:

- JuMP uses a generic solver-independent interface provided by the MathOptInterface package, making it easy to change between a number of open-source and commercial optimization software packages ("solvers").
- Currently supported solvers include Artelys Knitro, Bonmin, Cbc, Clp, Couenne, CPLEX, ECOS, FICO Xpress, GLPK, Gurobi, Ipopt, MOSEK, NLopt, and SCS.

Access to advanced algorithmic techniques:

- Including efficient LP re-solves which previously required using solver-specific and/or low-level C++ libraries.

Ease of embedding:

- JuMP itself is written purely in Julia. Solvers are the only binary dependencies.
- JuMP is MPL licensed, meaning that it can be embedded in commercial software that complies with the terms of the license.
- Being embedded in a general-purpose programming language makes it easy to solve optimization problems as part of a larger workflow (e.g., inside a simulation, behind a web server, or as a subproblem in a decomposition algorithm).
    - As a trade-off, JuMP's syntax is constrained by the syntax available in Julia.

Solvers:

JuMP depends on solvers to solve optimization problems. Most solvers are not written in Julia, and some require commercial licenses to use, so installation is often more complex.

Solver	Julia Package	License	Supports
Cbc	Cbc.jl	EPL	MILP
Clp	Clp.jl	EPL	LP
CPLEX	CPLEX.jl	Comm.	LP, MILP, SOCP, MISOCP
CSDP	CSDP.jl	EPL	LP, SDP
ECOS	ECOS.jl	GPL	LP, SOCP
FICO Xpress	Xpress.jl	Comm.	LP, MILP, SOCP, MISOCP
GLPK	GLPK.jl	GPL	LP, MILP
Gurobi	Gurobi.jl	Comm.	LP, MILP, SOCP, MISOCP
Ipopt	Ipopt.jl	EPL	LP, QP, NLP
MOSEK	MathOptInterfaceMosek.jl	Comm.	LP, MILP, SOCP, MISOCP, SDP
OSQP	OSQP.jl	Apache	LP, QP
SCS	SCS.jl	MIT	LP, SOCP, SDP
SeDuMi	SeDuMi.jl	GPL	LP, SOCP, SDP

Where:
LP = Linear programming
QP = Quadratic programming
SOCP = Second-order conic programming (including problems with convex quadratic constraints and/or objective)
MILP = Mixed-integer linear programming
NLP = Nonlinear programming
MINLP = Mixed-integer nonlinear programming
SDP = Semidefinite programming
MISDP = Mixed-integer semidefinite programming

Three advantages of using JuMP for solving Optimization problems over general purpose programming languages:

1 - User friendliness: Syntax that mimics natural mathematical expressions.

2 - Speed: Benchmarking has shown that JuMP can create problems at similar speeds to special-purpose modeling languages such as AMPL.

3 - Access to advanced algorithmic techniques: Including efficient LP re-solves which previously required using solver-specific and/or low-level C++ libraries.