OperationsResearchModels.jl: A Julia package for operations research models

Mehmet Hakan Satman [1]

1 Department of Econometrics, Istanbul University, Turkiye

DOI: 10.21105/joss.08592

Software

- Review
- Repository
- Archive

Editor: Arfon Smith

Reviewers:

- @odow
- @dourouc05

Submitted: 18 June 2025
Published: 02 September 2025

Summary

`OperationsResearchModels.jl` is a Julia package (Bezanson et al., 2017) that offers comprehensive implementations for numerous topics typically covered in an Operations Research (OR) curriculum. Its primary objective during development was to serve academic and pedagogical purposes, providing a clear and accessible platform for learning and applying OR concepts. While not optimized for high-performance computing, the package leverages JuMP for its underlying mathematical modeling, which inherently provides a reasonable level of computational efficiency. This design allows the package to deliver a suite of functions that solve classical operations research problems with remarkable ease and consistency, simplifying the process for students and researchers alike.

State of the field

Gurobi OptiMods is an open-source Python package that provides pre-implemented optimization use cases built on the Gurobi solver. Each module includes comprehensive documentation detailing its application and the underlying mathematical model (Gurobi Optimization, LLC, 2023). Julia's `Graphs.jl` (Fairbanks et al., 2021) package provides efficient methods for important network analysis topics such as minimal spanning tree and the shortest path. Google's OR-Tools is an open-source suite for solving optimization problems of Operations Research area. It provides a unified interface to various high-performance solvers and offers numerous examples for a wide range of applications, including vehicle routing, scheduling, and network flow. This makes it a versatile platform for researchers and practitioners to develop and implement advanced optimization solutions (Perron & Furnon, 2025). Although many software packages offer solvers for optimization, a comprehensive and educational data-driven solution is not readily available, at least for the Julia language.

`Julia Programming for Operations Research` (Kwon, 2019), a pioneering book in its field, serves as a testament to the fact that most topics in Operations Research courses can be addressed using only the Julia language, the JuMP (Lubin et al., 2023) package, and a selection of solvers. The `OperationsResearch.jl` package, in turn, provides a data-driven approach to this tooling.

Statement of Need

JuMP (Lubin et al., 2023) provides an excellent interface and macros for uniformly accessing optimizer functionality. Any mathematical optimization problem can be assembled with three core components: the objective function (`@objective`), variable definitions (`@variable`), and constraints (`@constraints`). The researcher's role is to formulate the original problem as a mathematical optimization problem and then to translate it using JuMP's macros.

OperationsResearchModels.jl streamlines the model translation stage by automatically constructing mathematical problems from problem-specific input data. Its extensive functionality encompasses a significant portion of the Operations Research domain. This includes, but is not limited to: the linear transportation problem, the assignment problem, the classical knapsack problem, various network models (Shortest path, maximum flow, minimum cost-flow, minimum spanning tree), project management techniques (CPM and PERT), the uncapacitated p-median problem for location selection, Johnson's Rule for flow-shop scheduling, a random key genetic algorithm for scheduling problems that are intractable to obtain a solution by Johnson's Rule, a zero-sum game solver, and a Simplex solver for real-valued decision variables.

Although the majority of computations are performed by the HiGHS optimizer (Huangfu & Hall, 2018) through JuMP, OperationsResearchModels.jl incorporates dedicated, hand-coded Simplex routines. These routines serve a valuable pedagogical purpose in Operations Research curricula, enabling users to observe and verify the detailed, step-by-step calculations involved in solving linear programming problems.

An Example

The example shown in **Table 1** defines a linear transportation problem with a given matrix of transportation costs between three sources and four targets, demand values of targets, and supply values of sources.

Table 1: Example transportation problem

	Target 1	Target 2	Target 3	Target 4	Supply
Source 1	1	5	7	8	100
Source 2	2	6	4	9	100
Source 3	3	10	11	12	200
Demand	100	100	100	100	

The Julia formulation of the problem is given below.

```
julia> problem = TransportationProblem(
            [
                1 5 7 8;
                2 6 4 9;
                3 10 11 12
            ],                    # Cost matrix
            [100, 100, 100, 100], # Demand vector
            [100, 100, 200],      # Supply vector
        )
```

The multiple-dispacthed `solve` function is responsible to solve many of the models implemented in the package. When the problem is in type of `TransportationProblem`, a special method called and the result is in type `TransportationResult`.

```
julia> result = solve(problem);
julia> result.cost
2400.0
julia> result.solution
[0.0 100.0 0.0 0.0; 0.0 0.0 100.0 0.0; 100.0 -0.0 -0.0 100.0]
```

The `solution` matrix represents the amounts sent from the sources to corresponding targets. **Table 2** represents the problem with its solution:

Table 2: Example transportation problem with its solution

	Target 1	Target 2	Target 3	Target 4	Supply
Source 1	1	5 (100)	7	8	100
Source 2	2	6	4 (100)	9	100
Source 3	3 (100)	10	11	12 (100)	200
Demand	100	100	100	100	

Acknowledgements

Our deepest appreciation goes to the Julia core and package developers for creating such a powerful computational tool and supportive environment.

References

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, *59*(1), 65–98. https://doi.org/10.1137/141000671

Fairbanks, J., Besançon, M., Simon, S., Hoffiman, J., Eubank, N., & Karpinski, S. (2021). *JuliaGraphs/Graphs.jl: An optimized graphs package for the Julia programming language*. https://github.com/JuliaGraphs/Graphs.jl/

Gurobi Optimization, LLC. (2023). *Gurobi OptiMods: Data-driven APIs for common optimization tasks*. https://github.com/Gurobi/gurobi-optimods.

Huangfu, Q., & Hall, J. J. (2018). Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, *10*(1), 119–142. https://doi.org/10.1007/s12532-017-0130-5

Kwon, C. (2019). *Julia programming for operations research*. Changhyun Kwon. ISBN: 9781798205471

Lubin, M., Dowson, O., Dias Garcia, J., Huchette, J., Legat, B., & Vielma, J. P. (2023). JuMP 1.0: Recent improvements to a modeling language for mathematical optimization. *Mathematical Programming Computation*. https://doi.org/10.1007/s12532-023-00239-3

Perron, L., & Furnon, V. (2025). *OR-tools* (Version v9.12). Google. https://developers.google.com/optimization/