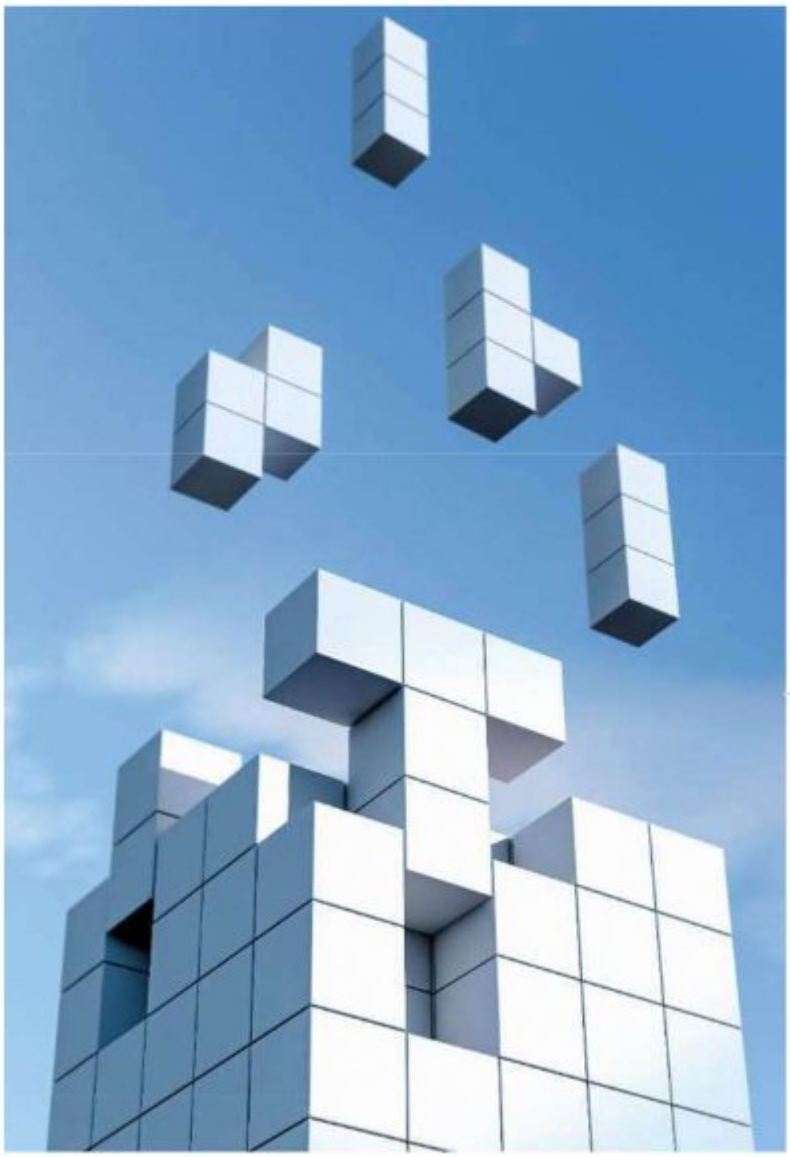
Tiger Analytics

SOLVE SMARTER
NOT HARDER

RETHINKING SOLVER STRATEGY FOR LARGE-SCALE OPTIMIZATION



An in-depth comparative study of open-source solver performance for analytics leaders building high-impact and agile decision systems

OUR WORK

1.1

OR PROBLEM FORMULATIONS

1.2

MATHEMATICAL ALGORITHMS TO SOLVE OR PROBLEMS

1.3

MATHEMATICAL ALGORITHMS VERSUS TYPICAL OR PROBLEMS

1.4

OPTIMIZATION SOLVERS VERSUS MATHEMATICAL ALGORITHMS

1.5

SOLVERS VERSUS OR PROBLEM TYPE

1.6

SOLVER ANALYSIS – LARGE-SCALE OPTIMIZATION

1.6.1

SOLVER ASSESSMENT - LINEAR PROGRAMMING

1.6.2

SOLVER ASSESSMENT – NON-LINEAR PROGRAMMING

1.6.3

SOLVER ASSESSMENT – FEW OTHER CALLOUTS

1.7

CONCLUSION

1.8

APPENDIX

THE PROBLEM-SOLVER FIT

When solving complex optimization use cases, such as yield maximization, waste minimization, and network optimization, analytics leaders are often required to move quickly. However, without a deep understanding of the underlying mathematical problem and the right solver strategy, the project could lead to cost inefficiencies and scalability challenges. This white paper examines how modeling choices and use-case-specific strategies help build reproducible and scalable optimization solutions and includes a comprehensive study on open-source solvers.

Our Work

From optimizing delivery routes in retail, scheduling manufacturing batches in consumer goods, allocating media spend in marketing, or setting dynamic prices in travel and hospitality, operational decisions are rarely straightforward. They involve a complex web of constraints, trade-offs, and changing business goals. **To remain competitive, sustainable, and profitable, industries must embrace real-time agility**.

Yet, scaling this agility is no small feat. The challenge for analytics leaders and operational researchers tasked with delivering these outcomes is translating business objectives into solvable mathematical models that take into account real-world constraints, changing inputs, and still provide timely, actionable outputs.

One approach to solving such nuanced and interconnected problems is through the lens of Large-Scale Optimization, a discipline within Operations Research (OR) which enables decision-making grounded in formal mathematical methods.

At Tiger Analytics, we have worked with **several Fortune 1000 clients across industries** on such problems over the years. We've observed that success with large-scale optimization often comes down to modeling choices and use-case specific strategies. Our experience across sectors has taught us which approach works best under different technical and operational conditions.

In practice, large-scale optimization often comes into play in production scheduling, capacity planning, inventory management, transportation and logistics, supplier selection, pricing optimization, and media planning – areas that are both high-impact and extremely complex from a business perspective. Mathematically modelling and solving such problems brings its own set of concerns.

Compute infrastructure requirements

Accounting for all variables that impact the decision

Achieving mathematically feasible/optimal solutions

Cost of enablement

In this white paper, we have summarized our learning, solving varied problems across multiple industries particularly focused on **our exploration of open-source and commercially licensed software, called Solvers.** The study provides the details of solvers available in the market, how they function, and their benefits and shortcomings.

The report is useful for data scientists and consultants who are solving large-scale OR problems, and on a regular basis make these decisions regarding which underlying solver to use.

The scope of this white paper covers the landscape of open-source solvers.

KEY TAKEAWAYS FROM THE RESEARCH

UNDERSTAND OPTIMIZATION PROBLEM COMPONENTS AND TYPES

The paper clearly defines the fundamental elements of an optimization problem, including objective functions (what to maximize or minimize), decision variables (what can be controlled), and constraints (limitations)

MAP OR PROBLEMS TO ALGORITHMS AND SOLVERS

The paper provides tabulations that link OR problem types to suitable mathematical algorithms (e.g., Simplex, Branch and Bound, Gradient) and then further connects these algorithms to specific open-source optimization solvers (e.g., GLOP, SCIP, IPOPT)

GATHER PERFORMANCE INSIGHTS FOR LARGE-SCALE OPTIMIZATION SOLVERS

The study rigorously assesses open-source solvers' performance in large-scale optimization for both Linear Programming and Nonlinear Programming Problem

IDENTIFY CONSIDERATIONS BEYOND PERFORMANCE

Beyond just execution time and memory, the paper also assesses solvers based on crucial practical aspects like Ease of Development, Convergence Issues

OR problem formulations

1.1

Before we get into the foundational aspects of OR problems, let's look at some typical **use cases** for which clients reach out to us

Production Scheduling

Optimizing the sequence and timing of manufacturing tasks to maximize output, minimize costs, and meet deadlines.

Capacity Planning

Determining the optimal production capacity or resource allocation to meet demand efficiently.

Inventory Management

Deciding on optimal inventory levels to balance storage costs, ordering costs, and the risk of stockouts.

Transportation and Logistics

Optimizing delivery routes, vehicle utilization, and network design to minimize transportation costs and improve delivery times. An example given is determining optimal transportation routes and quantities for minimizing shipping costs while satisfying demand and supply constraints

Let's have a look at some typical OR problems types

PROBLEM TYPE	DESCRIPTION	SAMPLE PROBLEM
LINEAR PROGRAMMING (LP)	This involves optimizing a linear objective function subject to linear constraints. The variables in LP are continuous.	Determining the optimal transportation routes and quantities for minimizing shipping costs while satisfying demand and supply constraints.
INTEGER PROGRAMMING (IP)	IP problems are similar to LP but require all decision variables to be integers (whole numbers).	Deciding the number of delivery trucks to deploy, where fractional trucks aren't feasible.
MIXED-INTEGER PROGRAMMING (MIP)	MIP problems combine the characteristics of both LP and IP, involving both continuous and integer decision variables.	Optimizing production scheduling where the number of machines is an integer, but production quantities are continuous.
QUADRATIC PROGRAMMING (QP)	QP problems involve an objective function that is quadratic and constraints that are linear or quadratic.	Minimizing costs in a manufacturing process that involves nonlinear relationships between production quantities and cost.
NONLINEAR PROGRAMMING (NLP)	NLP problems involve nonlinear objective functions or constraints. These are more complex and typically require specialized algorithms to solve.	Optimizing supply chain performance where transportation costs are nonlinear (e.g., bulk discounts based on the quantity).
CONSTRAINT PROGRAMMING (CP)	CP is focused on finding solutions to problems by satisfying a set of constraints often involving discrete variables.	Scheduling deliveries where various time windows, vehicle capacities, and delivery constraints need to be met.

Mathematical algorithms to solve OR problems

1.2

There are several mathematical models that could be used to solve a particular type of OR problem. We have restricted ourselves to the most typically used mathematical algorithms in the industry.

01

Simplex method:
A popular method for solving linear programming (LP) problems by iterating through feasible region vertices to find optimal solutions. Variants like the dual Simplex method handle infeasible starting solutions.

02

Branch and Bound method:
An algorithm for integer programming that explores a decision tree, dividing problems into subproblems ("branches") and using bounds to eliminate suboptimal solutions.

03

Branch and Cut method:
An algorithm for integer programming (IP) and mixed-integer programming (MIP) that combines branch and bound with cutting planes for improved efficiency.

04

Gradient method:
An iterative optimization algorithm that minimizes or maximizes a function by moving in the opposite direction of the gradient, commonly used in continuous optimization.

05

Interior Point method:
Algorithms for large-scale linear and nonlinear optimization problems that move through the interior of the feasible region, efficient for problems with many variables.

Mathematical algorithms versus typical OR problems

1.3

Here is a cross-tab between OR problems and mathematical algorithms as explained in previous sections

PROBLEM TYPE	SIMPLEX & VARIANTS	BRANCH AND CUT	BRAND AND BOUND	GRADIENT	INTERIOR POINT
Linear programming (LP)	x		x	x	x
Integer programming (IP)		x	x		
Mixed integer programming (MIP)		x	x		
Quadratic programming (QP)				x	x
Non-linear programming (NLP)				x	x
Constraint programming (CP)			x		

Table 1: OR problem type versus mathematical algorithms

Optimization solvers versus mathematical algorithms

1.4

The next step is to look at which solver supports these mathematical algorithms. The solvers listed below support various mathematical algorithms, but we have limited our selection to the ones that are most commonly used.

SOLVERS	SOLVER TYPE	SIMPLEX & VARIANTS	BRANCH AND CUT	BRAND AND BOUND	GRADIENT	INTERIOR POINT
GLOP	Open IP license	X				X
PDLP	Open IP license				X	X
SCIP	Open IP license	X	X	X		
CP-SAT	Open IP license			X		
CBC	Open IP license		X			
GLPK	Open IP license	X		X		X
HiGHS	Open IP license	X		X	X	X
IPOPT	Open IP license					X
BONMIN	Open IP license			X		
COUENNE	Open IP license			X		
SLSQP	Open IP license				X	
GUROBI	Commercial license	X	X	X		X
CPLEX	Commercial license	X	X	X		X
XPRESS	Commercial license	X	X	X		X
HEXALY	Commercial license	X	X	X		X

Table 2: OR problem type versus mathematical algorithms

Solvers versus OR problem type

1.5

Using the data from the two tables listed above, we can now synthesize a problem type versus solver matrix. This serves as a guide to data science practitioners starting on new OR use cases.

SOLVERS	LP	IP	MIP	QP	NLP	CP
GLOP	X					
PDLP	X	X	X		X	
SCIP	X	X	X		X	X
CP-SAT						X
CBC	X	X	X			
GLPK	X	X	X			

SOLVERS	LP	IP	MIP	QP	NLP	CP
HiGHS	X		X			
IPOPT					X	
BONMIN		X	X		X	
COUENNE		X	X		X	
SLSQP				X	X	
GUROBI	X	X	X	X	X	
CPLEX	X	X	X	X	X	
XPRESS	X	X	X	X	X	
HEXALY	X	X	X	X	X	

Table 3: OR problem type versus mathematical algorithms

Solver Analysis – Large-scale optimization

1.6

As referenced from Table 3, there are multiple solvers that can be used to solve a particular OR problem type. We tested these solvers against two problems to better assess their performance.

These are the two problems we worked on.

1. MARITIME INVENTORY ROUTING PROBLEM

Mixed Integer Programming Problem

2. SATELLITE LAUNCH VEHICLE PLANNING PROBLEM

Non-linear Programming Problem

Testing process

We evaluated each solver by progressively increasing the number of variables and measuring the following metrics across multiple runs using the same configuration.

Average execution (in seconds)

Average peak memory usage (in MBs)

Additionally, we also tested the reproducibility of the results across repeated runs.

1.6.1

USE CASE #1: MARITIME INVENTORY ROUTING PROBLEM (MIRP)

BRIEF

As the name suggests, this a combination of inventory management and routing problems. While MIRP is a class of problems, our focus for this study is a particular type called a single product maritime inventory routing problem. The key assumption here is that the time taken to travel across ports is significantly greater than the time a vessel spends at the port for loading/unloading. Therefore, on-port operations are not modelled here.

OBJECTIVE

Maximization of revenue, with revenue described as:

Revenue = Revenue from discharge - Travel cost - Attempt cost

CONSTRAINTS

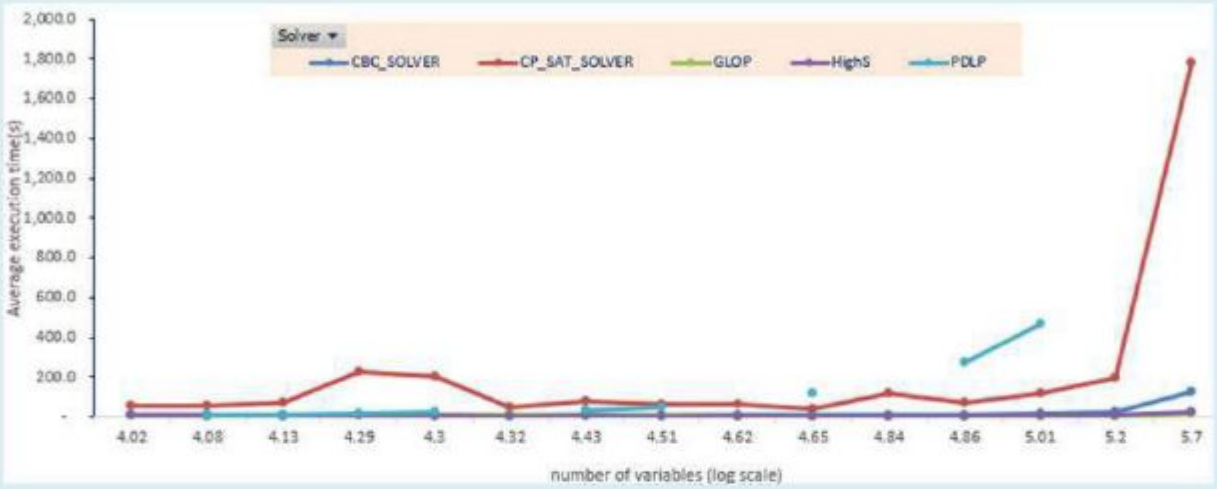
- Port flow balance constraint
- Inventory balance constraint for port and vessels
- Empty-return constraint to terminal port
- Travel-at-capacity constraint from origin port
- Market-based constraints on buy/sell limits with spot or open markets
- Port capacity constraint with regards to number of vessels that can dock and load/unload simultaneously

Refer the appendix for relevant literature

We tested solvers by increasing the number of decision variables from ~10k to ~500k and recorded the performance metrics for each configuration.

1.6.1.1 Average execution time versus number of variables

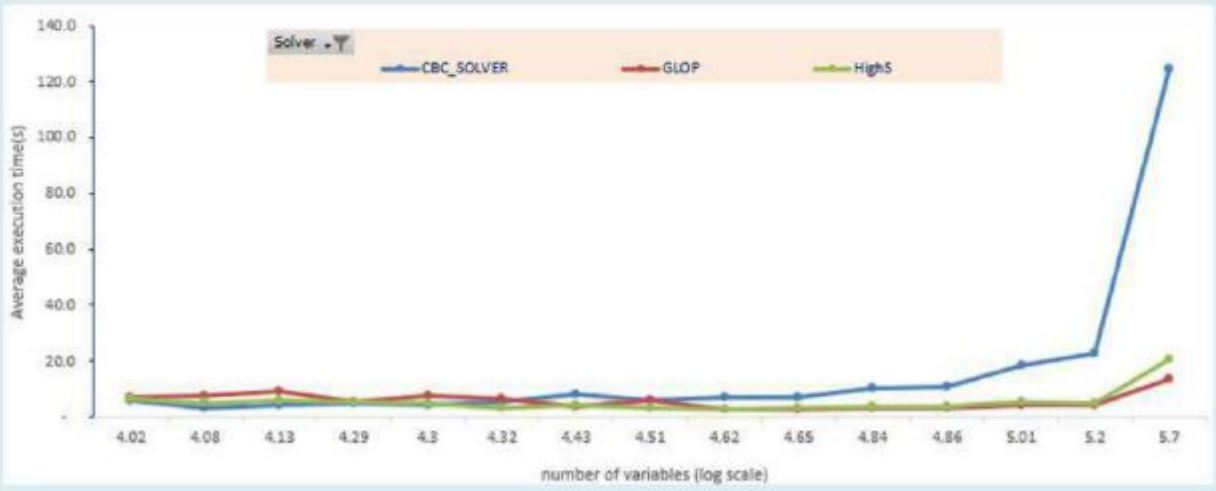
- **CP-SAT** behaves quite differently when compared to other solvers, which is expected given it is more of a constraint programming solver.
- **PDLP**, on the other hand, shows a sharp increase in execution time as we move from **10k to 100k** decision variables.



Plot 1.1: LP Solvers All – Average execution time versus number of decision variables

We get a better view of the other solvers after filtering out **CP-SAT** and **PDLP.**

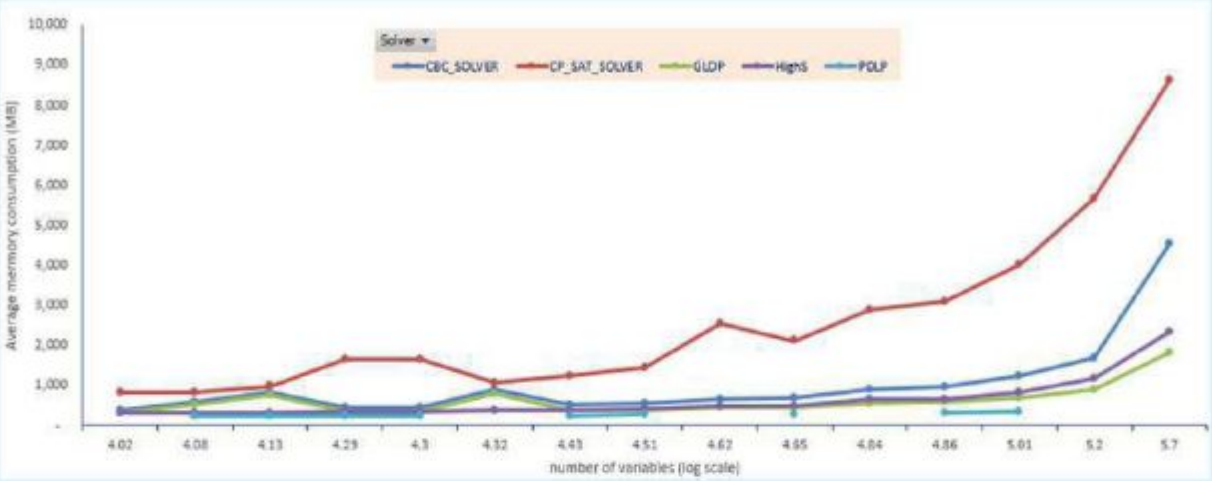
- As we move towards **50k** decision variables, **CBC** performance takes a hit.
- **HiGHS** and **GLOP** solvers continue to perform well throughout, though we can observe a slight increase in execution time as we move from **150k** decision variables to **500k** decision variables.



Plot 1.2: LP Solvers Select – Average execution time versus number of decision variables

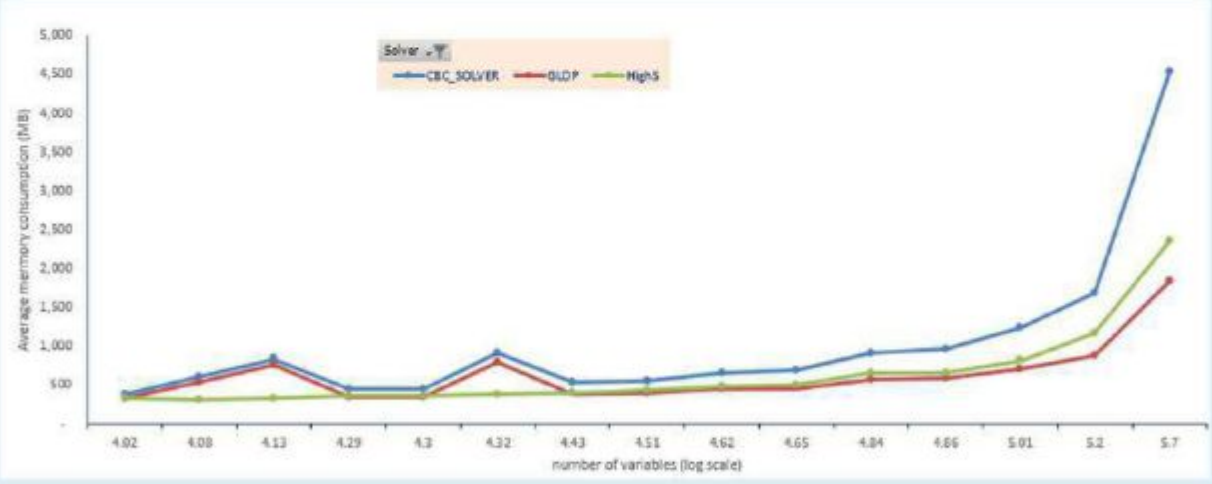
1.6.1.2 Average peak memory usage versus number of decision variables

- We see a quadratic increase in memory usage as the number of variables increase.
- As expected, **CP-SAT** consumes significantly more compute resources compared to other solvers.



Plot 2.1: LP Solvers All – Average peak memory usage versus number of decision variables

- Once we filter out **CP-SAT** and **PDLP** (due to insufficient data points), we notice:
- A significant impact on peak memory usage as we move from **~150k to ~500k** across all solvers.
- This impact is more exaggerated for **CBC** when compared to other solvers.



Plot 2.2: LP Solvers Select – Average peak memory usage versus number of decision variables

1.6.1.3 Reproducibility of results

When comparing solver outputs (objective function values) across multiple runs for the same configuration, we observed consistent results for all solvers each time. This indicates that the solvers produce reproducible outcomes under identical configurations.

1.6.2

USE CASE #2 SATELLITE LAUNCH VEHICLE PLANNING PROBLEM

BRIEF

Aerospace stands out as one of the most advanced and rapidly evolving fields in modern engineering, but it is also a capital-intensive industry. Hence, a lot of cost models have been developed to establish a relationship between launch vehicle development, deployment and the actual launch. Here, we're working on optimizing such cost models.

OBJECTIVE

Minimization of cost. We estimate costs using Cost Estimating Relationships (CERs) which quantify the correlation between different variables that impact launch vehicles and their associated cost components.

CONSTRAINTS

There are various constraints applied, which can be broadly classified into:

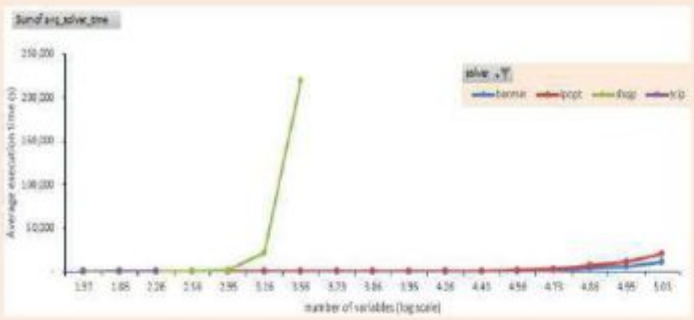
Constraints on interrelationship of variables

Constraints related to basic functioning, driven by scientific laws

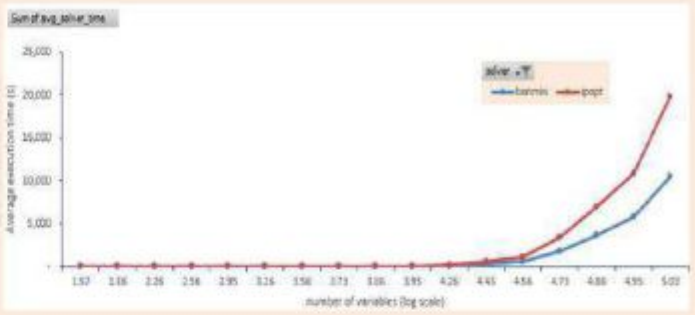
Constraints on overall performance

1.6.2.1 Average execution time versus number of variables

- Performance of both BONMIN and IPOPT solvers take a hit as we move towards more than 50k decision variables in the model.

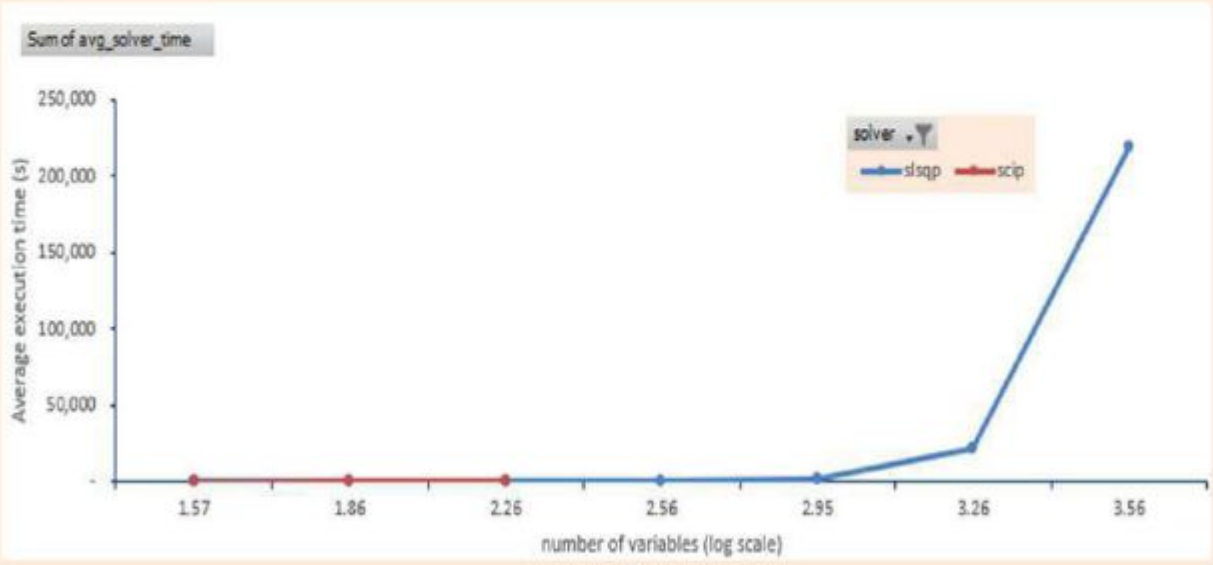


Plot 3.1: NLP Solvers All – Average execution time versus number of decision variables



Plot 3.2: NLP Solvers Select – Average execution time versus number of decision variables

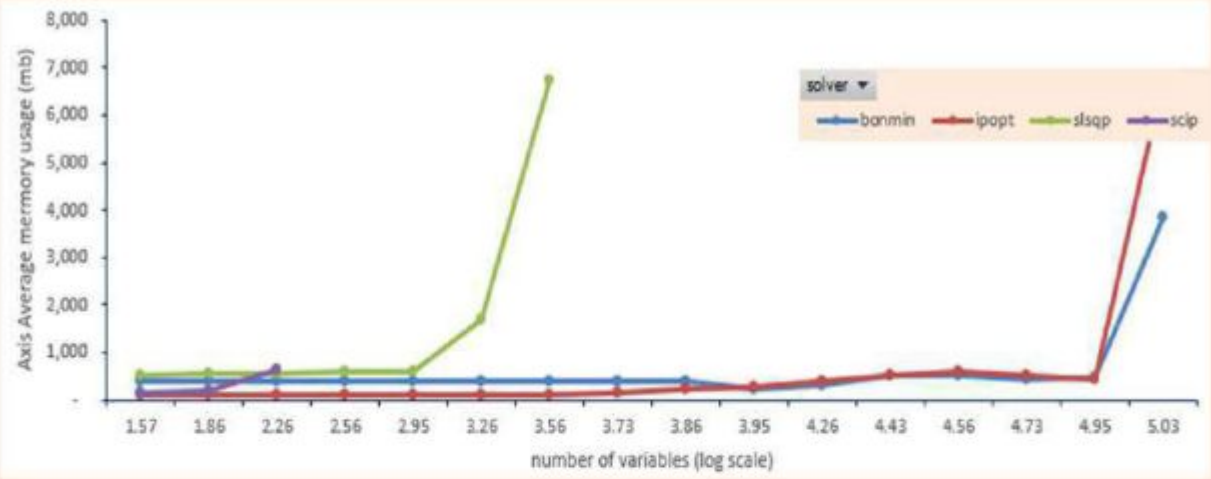
- **SCIP and SLSQP**, on the other hand, couldn't cope as we scaled the number of launch vehicles (i.e.) the number of decision variables
    - For **SCIP** solver with launch vehicle count between 1 and 5, its objective function value was exactly the same as **IPOPT** and **BONMIN**. However, as we scaled further, the solver ran indefinitely.
    - For **SLSQP**, objective function values were different from optimal values and were largely dependent on the initial states.



Plot 3.3: NLP Solvers Select– Average execution time versus number of decision variables

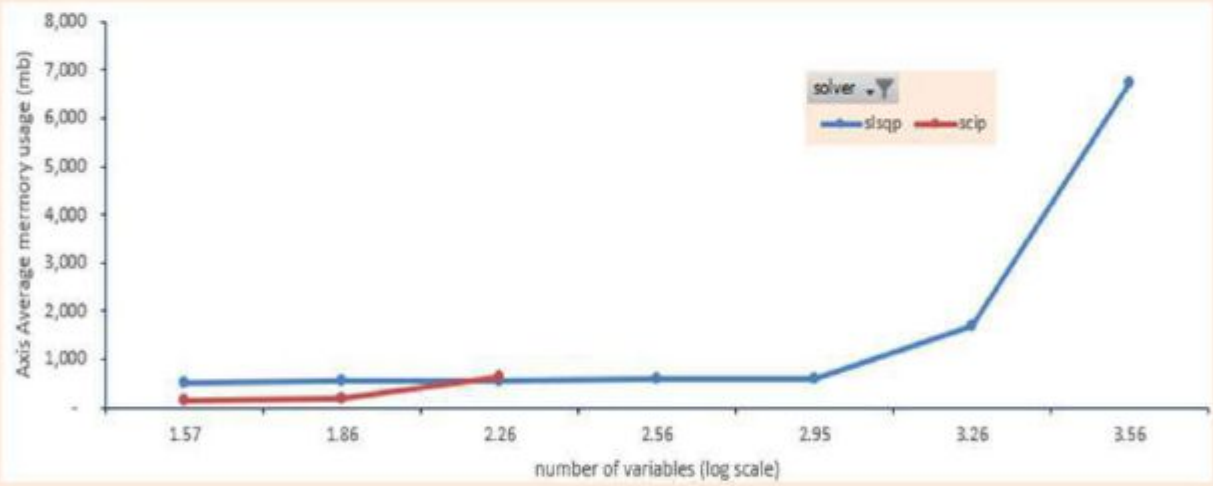
1.6.2.2 Average peak memory usage versus number of decision variables

- Memory usage for **IPOPT** and **BONMIN** solvers increases as we go beyond **10k** variables and jumps again only once we move to **~100k** variables



Plot 4.1: NLP Solvers All – Average peak memory usage versus number of decision variables

- Memory usage for **SLSQP** increases exponentially as we move towards **~100** variables



Plot 4.2: NLP Solvers Select – Average peak memory usage versus number of decision variables

For **SCIP**, tests with higher variable counts were discontinued as the solver exhibited indefinite runtimes, requiring manual termination.

FRAMEWORK	SOLVER	EASE OF DEVELOPMENT	CONVERGENCE ISSUES	BINARY VARIABLE SUPPORT	COMMUNITY SUPPORT	INTERPRETABILITY OF RESULTS
OR-Tools	**GLOP**	Easy	Low	Yes	Strong	High
OR-Tools	**PDLP**	Easy	Moderate	Yes	Strong	High
OR-Tools	**SCIP**	Easy	Moderate	Yes	Strong	High
OR-Tools	**CP-SAT**	Easy	Low	Yes	Strong	High
PuLP	**CBC**	Moderate	Moderate	Yes	Strong	High
HiGHS	**HiGHS**	Easy	Low	Yes	Strong	High
COIN	**Ipopt**	Moderate	Moderate	No	Moderate	Moderate
COIN	**Bonmin**	Moderate	Moderate	No	Moderate	Moderate
COIN	**Couenne**	Moderate	Moderate	No	Moderate	Moderate
COIN	**SLSQP**	Low	Moderate	No	Moderate	Moderate

Table 4: Efficiency highlights of solvers for OR applications

Conclusion

As enterprises continue to refine and accelerate their decision-making processes, large-scale optimization will play an increasingly crucial role in enabling real-time agility. Our study of open-source solvers highlights the importance of choosing the right solver strategy based on problem structure, performance needs and resource considerations. An understanding of the capabilities of different solvers empowers analytics leaders to make informed, context-aware decisions.

OUR RECOMMENDATIONS

PRIORITIZE HIGHS AND GLOP FOR LARGE-SCALE LP

For large-scale Linear Programming problems where performance is critical, **HiGHS** and **GLOP** appear to be robust choices, maintaining good performance even as variables increase

CONSIDER BONMIN/IPOPT FOR SCALABLE NLP

For large-scale Linear Programming problems where performance is critical, **HiGHS** and **GLOP** appear to be robust choices, maintaining good performance even as variables increase

EVALUATE SCIP AND SLSQP CAREFULLY FOR NLP

We recommend careful evaluation or avoidance of **SCIP** and **SLSQP** for very large-scale NLP problems due to observed indefinite runtimes (SCIP) and dependence on initial states/suboptimal results (SLSQP)

HOLISTIC EVALUATION

We strongly recommend using the data from Table 4 as a checklist when selecting a solver to ensure that factors such as ease of development, convergence reliability, binary variable support, and community assistance align with project needs and team capabilities.

Maritime Inventory Routing Problem:

Problem overview and data:
https://mirplib.scl.gatech.edu/home

Problem formulation:
https://mirplib.scl.gatech.edu/sites/default/files/Group1_MIP_Model.pdf

Authors

1.9

MOHAMMAD ZAMEER ABBAS – ASSOCIATE DIRECTOR, DATA SCIENCE

Abbas is a seasoned Data Science professional with about a decade of experience in the Data Science and Machine Learning space. Throughout his career, he has worked across various domains like CPG (Retail), Manufacturing, Pharmaceuticals, and Real Estate. In recent years he has been focusing on challenges related to Supply Chain and Manufacturing, Optimization (OR), along with Customer and Consumer relations.

AMIT JOSHI – DATA SCIENTIST

Joshi is a Data Scientist with experience in Marketing Analytics, Machine Learning, and Generative AI. He has worked with Fortune 100 clients in Consumer Packaged Goods (CPG), Retail, and Insurance domains. Along with working on key client engagements, he actively works on Optimization (OR) related R&D work.

Special thanks to our other contributors and guides:

- Nitesh Agarwal – Data Scientist
- Varadarajan Kousikan – Lead Data Scientist
- Suryateja Mudimi – Associate Director, Data Science
- Aarti Kapur – Partner, Data Science

ABOUT TIGER ANALYTICS

Tiger Analytics is a global leader in AI and analytics, helping Fortune 1000 companies solve their toughest challenges. We offer full-stack AI and analytics services and solutions to help businesses achieve real outcomes and value at scale. We are on a mission to push the boundaries of what AI and analytics can do to help enterprises navigate uncertainty and move forward decisively. Our purpose is to provide certainty to shape a better tomorrow

Being a recipient of multiple industry awards and recognitions, we have 4000 technologists and consultants, working from multiple cities in 5 continents.

www.tigeranalytics.com