University of North Carolina Asheville
Journal of Undergraduate Research
Asheville, North Carolina
May 2021

Downstream: An Open-Source Vehicle Routing Solution

Jesse Reeve
Department of Computer Science
The University of North Carolina Asheville
One University Heights
Asheville, North Carolina 28804 USA

Faculty Advisor: Dr. Kevin Sanft

Abstract

Optimizing vehicle routing is an essential problem of the global economy that has engaged computer scientists since the 1950s. Today, the technology for solving these optimization problems is well established; but algorithm-based vehicle routing is primarily available through subscription-based "software as a service" models developed for large organizations. The Downstream project explores the potential of stand-alone, open source software to support smaller organizations with free or low-cost vehicle routing. FIRST at Blue Ridge is one such small organization: a nonprofit, residential 12-step recovery center in Ridgecrest, NC. FIRST's clients depend on reliable transportation scheduling to engage with the community, but FIRST's limited budget and technical staff has prevented them from exploring algorithmic vehicle routing. Downstream is a user friendly, open-source software package that fills the gap revealed by FIRST's needs. Using the cut-and-branch algorithm devised by Cordeau (2006), Downstream uses linear programming techniques to generate daily transportation schedules optimized to reduce wait time and wear and tear on vehicles. Downstream's tech stack is accessible for free at a reasonable level of use, including spreadsheet software such as Microsoft Excel, the Google Cloud API and Google's OR-Tools optimization suite. The software is designed to be easy for a first-time computer user to install and run, with a simple step-by-step installation process. Replacing FIRST's manual routing process with Downstream saves hours of human effort every week, reduces vehicle mileage, and prevents unnecessary wait times for clients. We hope that other small organizations can reap similar benefits.

1. Introduction

We undertook the Downstream project on the hypothesis that a software-as-product approach to transportation routing could provide significant value to small organizations like FIRST. While Downstream draws on a deep pool of existing research and development, most commercially available routing solutions use a software-as-service approach, entailing subscription fees that impose a significant financial burden on a small nonprofit like FIRST. Relying on an outside service also creates potential problems with continuity of service and confidentiality of client information. Downstream offers a lightweight, mission-focused alternative to these existing services.

At the most general level, vehicle routing problems (VRPs) are computationally intractable. That said, route optimization is a critical problem in industrial logistics, a multi-trillion dollar global market. Accordingly, many algorithmic and heuristic approaches have been developed to attack subsets of VRPs defined by constraints. FIRST's transportation needs correspond to the Dial-A-Ride Problem (DARP), which is itself well-studied due to its widespread practical application in real-world scenarios. Downstream builds on the existing body of work on the DARP to create an approachable software product for a non-technical user.

2. Literature Review

Because route optimization is so important to industrial logistics, it is an extremely well-studied field. Most approaches begin with the Vehicle Routing Problem (VRP) developed by Dantzig and Ramser[1] in 1959, one of the first commercial applications of computers. The VRP is an NP-hard problem[2], but can be made more tractable by adding constraints that further specify the particular problem at hand, creating a taxonomy of problems that admit different solution procedures. FIRST's transportation needs closely resemble the Dial-A-Ride Problem, which generates routes and schedules for users who request pickup at an origin location and dropoff at a destination location. In graph theory terms, each user's requested trip represents an edge between an origin and destination node. *Vehicle Routing*[3], a joint publication of The Society for Industrial and Applied Mathematics and the Mathematical Optimization Society, provides an overview of the VRP With Time Windows in chapter 5, and the DARP in chapter 7.

Once the appropriate constraints have been added to the general VRP, the practice originated by Dantzig and Ramser[1] is to solve the resulting problem using linear programming (LP), which finds the optimum value of a linear function given linear constraints in the form of equations or inequalities. For the sake of the DARP, the function to be optimized ("objective function") represents the total travel distance or time by the vehicle fleet; the constraints describe the parameters of the problem. For instance, a constraint may require a vehicle to drop off a given passenger only after that passenger is picked up, or require the vehicle that drops off a passenger to be the same vehicle that picked the passenger up. The resulting LP problem can be solved by any of a variety of open-source or commercially available LP solvers.

After reviewing the discussion of the DARP in *Vehicle Routing* and "Survey of Dial-A-Ride Problems"[4] by Ho et al., we selected Jean-François Cordeau's algorithm described in "A Branch-and-Cut Algorithm for the Dial-A-Ride Problem"[5]. The next step was to formalize the constraints that defined an acceptable solution. Based on the needs of FIRST, we decided that minimizing delay was a higher priority than minimizing travel distance. Orda and Rom's "Shortest-Path and Minimum-Delay Algorithms in Networks with Time-Dependent Edge-Length"[6] provided insight into the mathematical requirements of balancing those constraints to match FIRST's priorities. We also reviewed Schilde and Hartl's "Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports"[7] in consideration of possibly providing Downstream with dynamic routing capability; however, we decided not to implement dynamic routing due to time constraints and the fit to the static routing process FIRST currently has in place.

3. Project Description

Downstream is an app designed to meet the transportation routing needs of FIRST at Blue Ridge. FIRST is a residential 12-step recovery program with around 150 clients who live on-campus. Almost all of those clients work, and every day brings many off-campus appointments as well. Without optimization software available, FIRST staff currently do all transportation routing, scheduling, and planning "by hand." This is a time-consuming and burdensome task that must be repeated every day, and due to confidentiality issues must be performed by administration-level staff.
Downstream closes that technological gap with powerful and accessible resources Google's OR-Tools optimization suite, Google Cloud services, and familiar office spreadsheet software. The combination of these freely-available products allows Downstream users to shift the burden of transportation scheduling to a fast, highly accurate software solution. Downstream's demonstrable value to FIRST at Blue Ridge indicates the opportunity for open-source software to provide great value to small organizations, particularly nonprofits and those with a limited budget.

3.1 Requirements/Specifications

Software requirements:
Spreadsheet software (e.g. Microsoft Excel, iWork Numbers, LibreOffice Calc, Google Sheets)

Hardware requirements:
Desktop PC or Mac
Windows 7 64-bit or Mac OS X 10.9.2 or later
Quad-core Intel or AMD processor, 2.5 GHz or faster

NVIDIA GeForce 470 GTX or AMD Radeon 6870 HD series card or higher
8 GB RAM

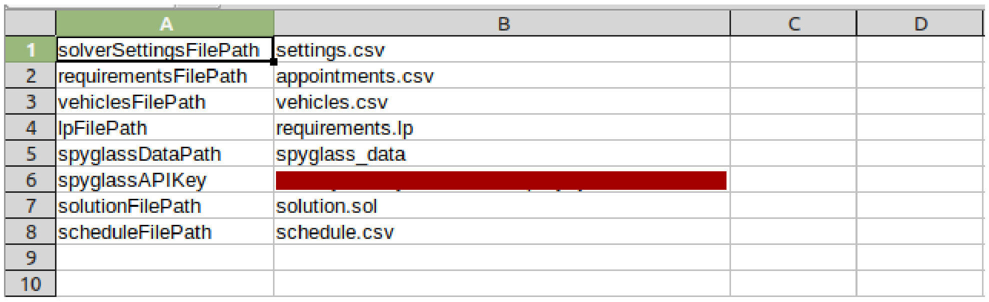
Other requirements:
 Internet access (No browser restrictions; Downstream includes its own HTTPS client)
 Free Google Cloud account (nonprofit upgrade optional)

3.2 Design

Downstream accepts user data from spreadsheet files and transforms it into an LP problem using Google's OR-Tools optimization suite. Once the OR-Tools solver finds a solution, Downstream parses it into a human-readable spreadsheet file. Because Downstream's users may have minimal familiarity with technology, operating Downstream is a matter of running a single executable file, turning input files into a ready-to-use timetable with one double-click.

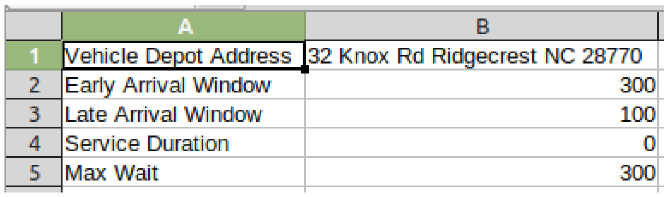
In addition to original code, Downstream incorporates several open-source resources. We credit Niels Lohmann for his JSON for Modern C++ library[8]; Ben Strasser for his Fast CSV Parser library[9]; GitHub user Yhirose for cpp-http lib[10]; and Google for their OR-Tools suite[11], including LP, MIP, and cut-and-branch solvers, as well as the Google Maps API[12].

The Downstream app incorporates four user input files: .config.csv, settings.csv, vehicles.csv, and appointments.csv. The .config file includes variables for file paths and the Google Cloud API key. Once the user provides their unique API key, these parameters need never change. The settings file includes configurable solver parameters such as the allowable early arrival window and the maximum client wait time, which can be tightened for a timely schedule or relaxed if no solution is otherwise possible. The vehicles file contains a list of vehicles in the organization's fleet and their passenger capacity. The day's transportation requirements are located in the appointments.csv file; each entry includes the client or clients served at an appointment, the number of clients being transported, the time window of the appointment, and the appointment address.



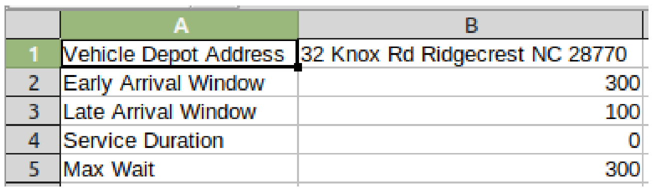
	A	B	C	D
1	solverSettingsFilePath	settings.csv		
2	requirementsFilePath	appointments.csv		
3	vehiclesFilePath	vehicles.csv		
4	lpFilePath	requirements.lp		
5	spyglassDataPath	spyglass_data		
6	spyglassAPIKey	[redacted]		
7	solutionFilePath	solution.sol		
8	scheduleFilePath	schedule.csv		
9				
10				

Example config.csv file, viewed in LibreOffice spreadsheet software. API key redacted.



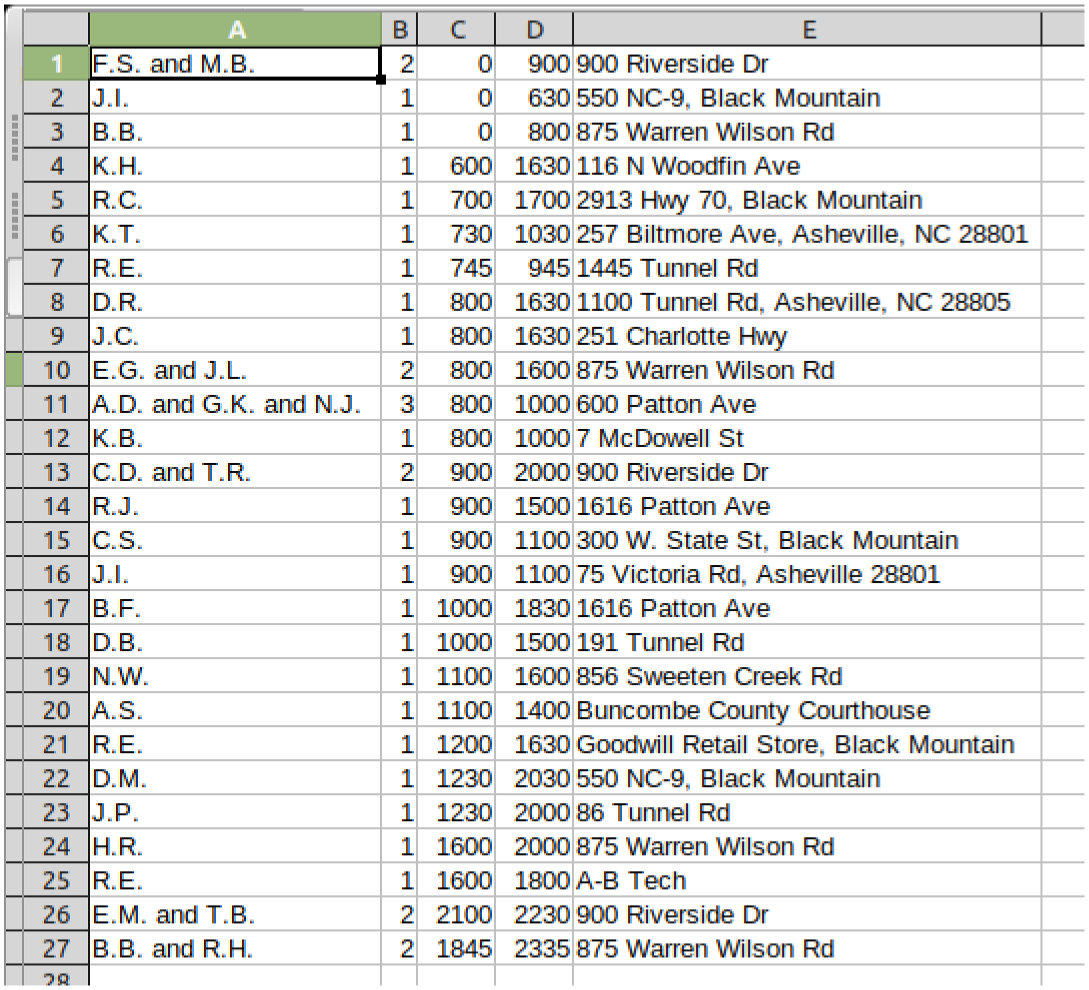
	A	B
1	Vehicle Depot Address	32 Knox Rd Ridgecrest NC 28770
2	Early Arrival Window	300
3	Late Arrival Window	100
4	Service Duration	0
5	Max Wait	300

Example settings.csv file, viewed in LibreOffice spreadsheet software.



	A	B
1	Vehicle Depot Address	32 Knox Rd Ridgecrest NC 28770
2	Early Arrival Window	300
3	Late Arrival Window	100
4	Service Duration	0
5	Max Wait	300

Example vehicles.csv file, viewed in LibreOffice spreadsheet software.



	A	B	C	D	E
1	F.S. and M.B.	2	0	900	900 Riverside Dr
2	J.I.	1	0	630	550 NC-9, Black Mountain
3	B.B.	1	0	800	875 Warren Wilson Rd
4	K.H.	1	600	1630	116 N Woodfin Ave
5	R.C.	1	700	1700	2913 Hwy 70, Black Mountain
6	K.T.	1	730	1030	257 Biltmore Ave, Asheville, NC 28801
7	R.E.	1	745	945	1445 Tunnel Rd
8	D.R.	1	800	1630	1100 Tunnel Rd, Asheville, NC 28805
9	J.C.	1	800	1630	251 Charlotte Hwy
10	E.G. and J.L.	2	800	1600	875 Warren Wilson Rd
11	A.D. and G.K. and N.J.	3	800	1000	600 Patton Ave
12	K.B.	1	800	1000	7 McDowell St
13	C.D. and T.R.	2	900	2000	900 Riverside Dr
14	R.J.	1	900	1500	1616 Patton Ave
15	C.S.	1	900	1100	300 W. State St, Black Mountain
16	J.I.	1	900	1100	75 Victoria Rd, Asheville 28801
17	B.F.	1	1000	1830	1616 Patton Ave
18	D.B.	1	1000	1500	191 Tunnel Rd
19	N.W.	1	1100	1600	856 Sweeten Creek Rd
20	A.S.	1	1100	1400	Buncombe County Courthouse
21	R.E.	1	1200	1630	Goodwill Retail Store, Black Mountain
22	D.M.	1	1230	2030	550 NC-9, Black Mountain
23	J.P.	1	1230	2000	86 Tunnel Rd
24	H.R.	1	1600	2000	875 Warren Wilson Rd
25	R.E.	1	1600	1800	A-B Tech
26	E.M. and T.B.	2	2100	2230	900 Riverside Dr
27	B.B. and R.H.	2	1845	2335	875 Warren Wilson Rd
28					

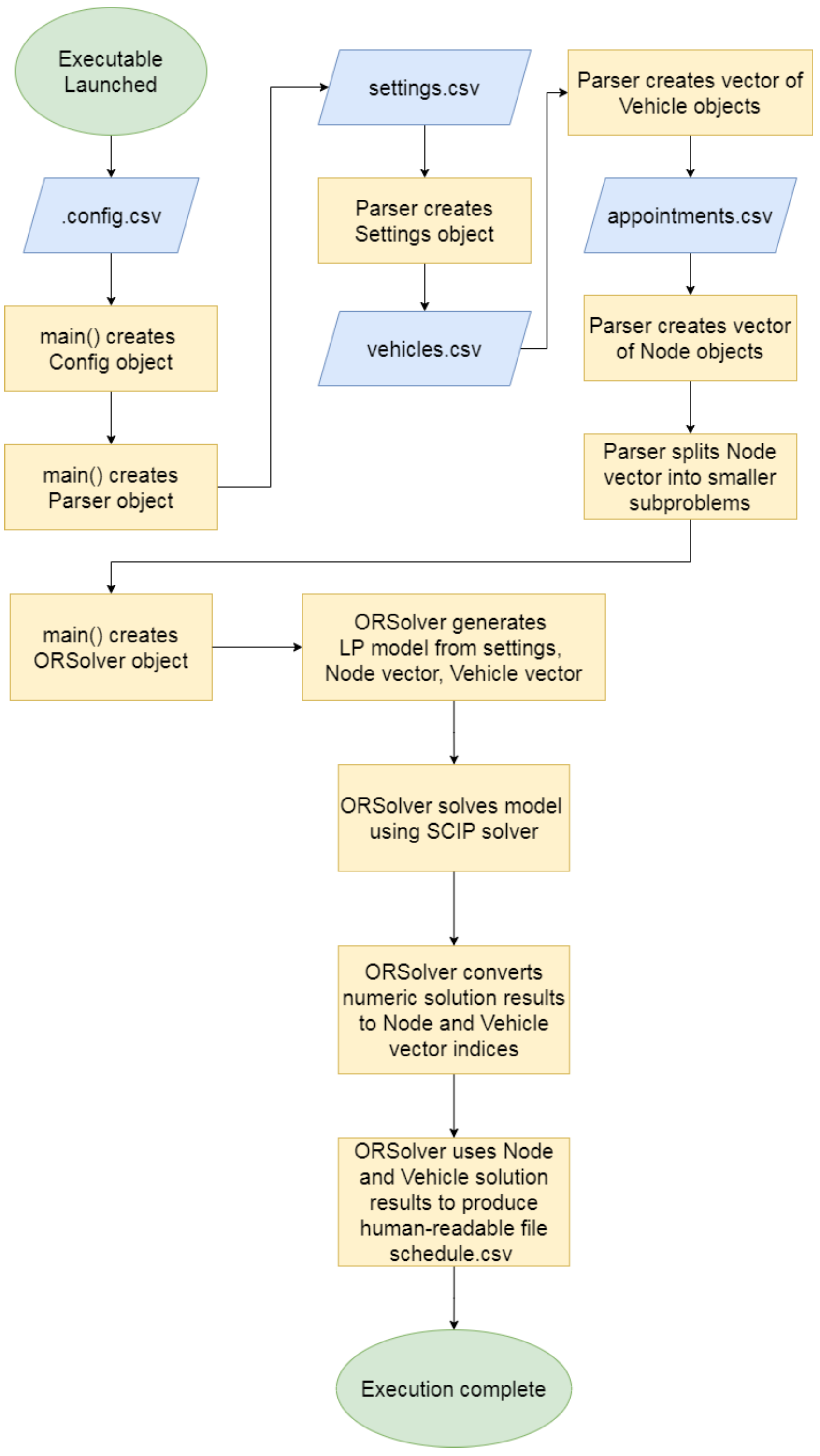
Example appointments.csv file, viewed in LibreOffice spreadsheet software.

After reading the data provided by the user, the Parser creates a vector of Node data structures representing pickup or dropoff from a single address. Each Node has data fields for passenger name(s), earliest and latest allowable service time, service duration, address, load, and a unique ID. A pickup Node has a positive load; a dropoff Node has a negative load, representing passengers leaving the vehicle. One appointment represents four Nodes: a pickup and dropoff Node for transportation to the appointment, and a pickup and dropoff Node for the return trip to FIRST. In order to manage problem size and increasing execution time, the Parser splits the list of Nodes into a series of smaller problems, each representing a maximum of ten appointments. See section 3.3 for further discussion of the execution time issue.

Once Downstream parses this information, its Spyglass module uses the Google Cloud API and an integrated HTTPS client to automatically and securely collect travel information. Spyglass uses the free Google Places API to standardize addresses and locations so that Downstream can recognize addresses despite minor changes in entry. For instance, the address “Karpen Hall, Asheville, NC” and “Karpen Hall, Asheville, North Carolina, 28804” are correctly identified as the same address. This reduces billed API calls and local storage.

After standardizing addresses, Spyglass acquires the travel time and distance for each pair of addresses using the Google Maps API. In order to adhere to the software-as-product model, Spyglass limits API calls to Google's free allotment by default. After downloading information from an API call, Spyglass stores it locally in JSON-formatted files so that the same call need not be made again. Thanks to Downstream's careful budgeting of billed API calls, it can acquire information for hundreds of addresses using the $200 monthly budget for free API calls that Google provides.

Using the client's requirements and route data provided by Spyglass, Downstream creates an ORSolver object that applies Cordeau's constraints to transform the collected information into a LP problem. The ORSolver object uses the OR Tools library's linear solver to solve the LP problem, then converts the numerical solution to addresses, vehicle IDs, and travel times, formatting a human-readable schedule in the file schedule.csv.


Executable Launched
.config.csv
main() creates Config object
main() creates Parser object
settings.csv
Parser creates Settings object
vehicles.csv
Parser creates vector of Vehicle objects
appointments.csv
Parser creates vector of Node objects
Parser splits Node vector into smaller subproblems
main() creates ORSolver object
ORSolver generates LP model from settings, Node vector, Vehicle vector
ORSolver solves model using SCIP solver
ORSolver converts numeric solution results to Node and Vehicle vector indices
ORSolver uses Node and Vehicle solution results to produce human-readable file schedule.csv
Execution complete

Process flow diagram for Downstream.

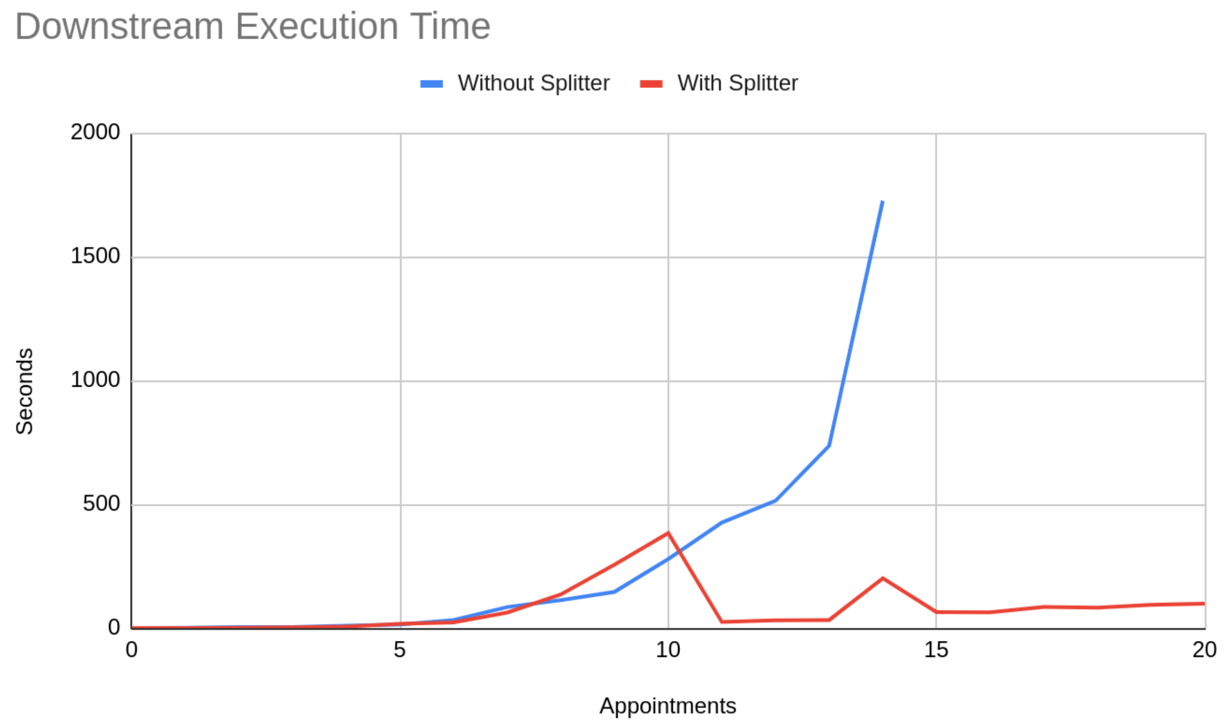
3.3 Testing, Validation, and Verification

To ensure that it yields accurate, efficient solutions, Downstream has been tested on edge cases, examples, and anonymized scheduling data provided by FIRST. We have also scheduled a "phase-in" period during which it will be used in tandem with FIRST's existing scheduling process to identify opportunities for improvement to both its algorithm and its user experience.

During the testing phase we encountered a significant issue with Cordeau's DARP algorithm: exponential growth. Cordeau's algorithm incorporates an objective function and several constraints that grow in O(n^2) relative to the number of nodes, where each appointment represents four nodes or two edges. Our original design used the COIN-OR optimization suite and wrote the optimization problem in .lp file format; however, we found that reproducing realistic transportation schedules as LP problems in .lp file format resulted in files too large for COIN-OR's interpreter to correctly parse. We then pivoted to using Google's OR Tools API to create the problem in memory. This circumvented the issue with COIN-OR's .lp interpreter, but the problem of exponential growth remained.

Cordeau's test cases go up to 32 "users" (64 nodes, or 16 appointments) in size, and five of his 30 test cases could not be solved to optimality in 12 hours of CPU time (21). A typical day at FIRST represents 20-30 appointments, each consisting of two edges (outgoing and ingoing), for an average of over 80 nodes. None of the one-day schedules provided by FIRST represented a problem small enough to be solved by Cordeau's algorithm in a feasible length of time.

In order to keep solution time within reasonable bounds, we incorporated a function into the parser that splits the appointments into "bite-sized" chunks of at most ten appointments. We then developed an automated script to run 20 tests on appointments.csv files randomly generated from FIRST input. Due to the rapid growth of execution time without the splitter function, we were unable to test problems larger than 14 appointments on the original Downstream algorithm; however, the execution time shows a clear exponential curve as appointments increase. The splitter algorithm keeps execution time under seven minutes in test cases up to 20 nodes in size.



Downstream problem size (appointments) vs. execution time. Each data point is the average execution time of 20 randomly generated problems.

4. Future Work

Our original testing plan included the development of a schedule evaluator, which would sum the travel time, vehicle mileage, and client wait time for a given schedule to allow quantitative comparison between multiple solutions for a given problem. Unfortunately the time allocated for the schedule developer was unexpectedly taken up by the pivot from COIN-OR's .lp interface to OR-Tools API.

The schedule evaluator software would also allow side-by-side testing of a wide variety of other solvers, solver parameters, heuristic approaches, and possibly even other LP formalizations of the DARP. The field of mathematical optimization is both wide and deep, and while we believe our research is well-targeted, it is limited by time and background knowledge.

In the shorter term, we do not intend to lose sight of our customer's interests. We will remain in contact with FIRST at Blue Ridge, helping them master Downstream and using their feedback to improve future versions of the software. Once FIRST has incorporated Downstream into their workflow, we will be able to produce non-technical written documentation and a short series of instructional videos for training and reference purposes.

While Downstream was developed according to the needs of FIRST at Blue Ridge, it was never intended to be solely for their benefit. We have elected to make Downstream available as open-source software in hopes that other organizations can make use of its capabilities or use it as a template for software that serves their differing needs.

5. Results and Reflection

Development of Downstream includes industry-wide best practices of requirements elicitation, goal-oriented development documents, utilization of third-party libraries, end-to-end testing, clean code style, and documentation for both maintenance and use. In creating Downstream, we focus on quality from the beginning of development to the last stages of deployment, from the standards of code creation to the rigor and customer value of the solvers' solutions. The resulting application is not only functional, but accessible to code improvement, customization, and additional features.

Downstream's business value is evidence-based and easy to demonstrate based on its software-as-product model and impact on the scheduling workload of FIRST administrative staff. With the ability to use multiple solvers, optimize solutions based on client requirements, and produce a printable list via a maximally user-friendly process, it is pinpointed to the needs of our client. Downstream evanesces a lengthy, burdensome process into a press of a virtual button.

One possible avenue for future work is developing Downstream as a web app. Downstream already requires and uses the internet to gather essential data, and all the required software to offer Downstream online is available in a variety of formats, both proprietary and open-source. Our choice to make Downstream a local app was driven by the priority that it be readily available and best serve FIRST, not any technical infeasibility of presenting it on the web.

We would particularly add that, given that Google offers not only Maps and Sheets spreadsheet software, but even its own open-source optimization suite and solver, OR-Tools and GLOP, all the pieces are in place for Google to develop a vehicle routing product like Downstream. We believe that such a product, backed by Google's technical expertise and support, could be a source of revenue or an invaluable contribution to the open-source software community.

However, the Downstream project itself demonstrates an opportunity for that open-source community to rise to the occasion, regardless of what for-profit corporations like Google may decide. Downstream is a single point in a vast space of optimization solver options, software dependencies, input and output configurations, and client preferences. From that vantage point, we are convinced that transportation scheduling is properly the work of software, not human effort. We hope that in the future, Downstream and other products like it will carry that load.

6. References

1. Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. Management Science, 6(1), pp. 80–92.

2. Toth, P. and Vigo, Daniele., eds. The Vehicle Routing Problem. Monographs on Discrete Mathematics and Applications. 9. Philadelphia: Society for Industrial and Applied Mathematics. 2002.

3. Society for Industrial and Applied Mathematics and Mathematical Optimization Society. Vehicle Routing: Problems, Methods, and Applications, edited by Toth, Paolo and Daniel Vigo. 2nd ed., Society for Industrial and Applied Mathematics, 2014.

4. Ho, Sin C., W.Y. Szeto, Yong-Hong Kuo, Janny M.Y. Leung, Matthew Petering, and Terence W.H. Tou. "A survey of dial-a-ride problems: Literature review and recent developments." Transportation Research, vol. 111, part B, 2018, pp. 395-421.

5. Cordeau, Jean-François. "A Branch-and-Cut Algorithm for the Dial-A-Ride Problem." Operations Research,vol. 54, no. 3, 2006, pp. 573-586.

6. Orda, Ariel and Raphael Rom. "Shortest-Path and Minimum-Delay Algorithms in Networks with Time-Dependent Edge-Length." Journal of the ACM, vol. 37, no. 3, 1990, pp. 607-625.

7. Schilde, M. K.F. Doerner, and R.F. Hartl. "Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports." Computers & Operations Research, vol. 38, 2011, pp. 1719-1730.

8. Lohmann, Niels. *JSON for Modern C++ version 3.9.1*. GitHub, 6 August 2020. https://github.com/nlohmann/json/

9. Strasser, Ben. *Fast C++ CSV Parser*. GitHub, 3 January 2021. https://github.com/ben-strasser/fast-cpp-csv-parser

10. Yhirose. *Cpp-httplib*. GitHub, 6 April 2021. https://github.com/yhirose/cpp-httplib

11. Google. *Cloud Maps Platform*. Google, 6 April 2021. https://developers.google.com/maps

12. Google. *OR-Tools*. Google, 6 April 2021. https://developers.google.com/optimization