Extensions to Robust Vehicle Routing

Brady Metherall

Having groceries delivered to one's door is becoming quite common. Tesco has over 700 delivery vans which deliver over 1000 orders on a typical day. Since household grocery orders are relatively small, delivery vans can hold many orders, and so, efficient routes must be found to make home delivery commercially viable. A consequence of the large number of deliveries per van is that the routes become much longer, and much more difficult to compute optimal routes due to the increased number of possibilities. Tesco allows their customers to choose a time window, typically 1-hour, in which they want their groceries delivered. This constraint gives an implicit ordering of the customers, which can be used to our advantage.

Mathematical Model

Initially, the time window constraint seems like it makes the problem more difficult. However, the implicit ordering of the customers greatly reduces the number of feasible solutions to test. In addition to customers selecting a 1-hour time window, they also have the option of selecting a 4-hour time window. This gives us the flexibility of assigning such a customer to a particular time window. Generally, only one or two customers per van have 4-hour time windows. Figure 1 shows a simplified vehicle routing problem with time windows.

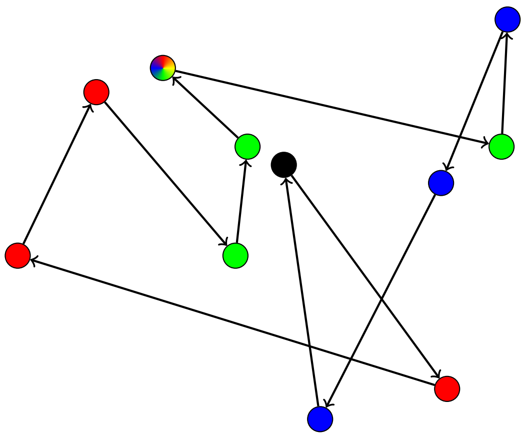

Figure 1—Example of a Tesco delivery van's route. The black node is the depot; red, green, and blue nodes are customers that specified a 1-hour time window; and the multi-coloured node is a customer that chose a 4-hour time window. The distance optimal route is shown.

To tackle the vehicle routing problem, we use the mathematical optimization and programming method dynamic programming. Dynamic programming is built on two defining characteristics, recursion and memoization, and guarantees we obtain the optimal solution. Dynamic programming splits a problem into sub-problems, which are then solved using recursion; memoization is the practice of storing results in a look-up table. This allows the result of a repeated, expensive computation to be looked up and reused, instead of recomputed from scratch, in this way, the program has a 'memory'. Although dynamic programming allows us to find the optimal solution, in the case of several 4-hour customers, actually computing the optimal solution is too computationally expensive. The number of combinations to test grows exponentially—to combat this, we consider two classes of heuristics to assign 4-hour customers to time windows. Furthermore, our algorithms are designed to be compatible with a variety of objective functions. This allows Tesco to use various objectives under different circumstances; some of the objectives include trip time, distance travelled, number and severity of late deliveries, and total standby time of customers. Objectives can be added together to form a more sophisticated objective as well.

Results

We show the timings for a variable number of time windows with four customers in each window in Figure 2. We are able to find the optimal route for 16–20 customers in only a millisecond or two. Interestingly, for the smaller systems the unmemoized version is faster than the memoized. This is actually to be expected—the memoized version has the additional overhead of searching and building the look-up table on each iteration. However, for the 7×4 system, the memoized version is close to 100 times faster.

Engineering and Physical Sciences Research Council

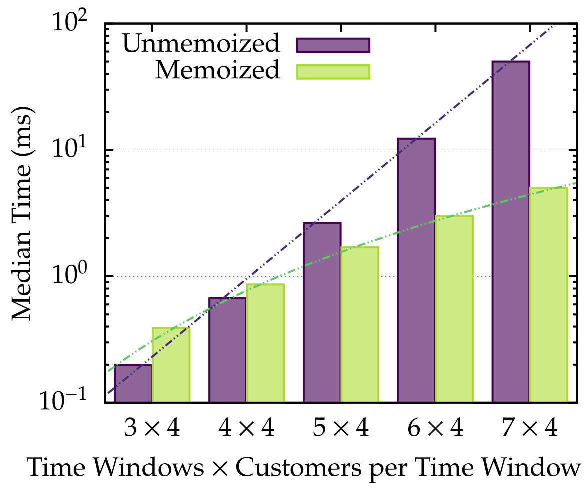

Figure 2—Computation time required to find the optimal vehicle route for a variable number of time windows.

Finally, we focus on the performance of our heuristics. Here, we highlight the case of four time windows of three customers each, with four additional 4-hour customers (Table 1). With four 4-hour customers the optimal solution requires testing all $4^4 = 256$ possibilities, whereas the heuristics intelligently allocates the 4-hour customers to a time window, and only tests this possibility. The optimal solution takes over 400ms, while the heuristics only require about 2.5ms—and very little memory. In this scenario, the heuristics find the optimal solution only 25% of the time, yet, provide a route only 2.6% longer than optimal on average, while being about 165 times faster and using 1/767 the amount of memory.

Conclusions

By taking a dynamic programming approach we were able to develop concise algorithms for solving the vehicle routing problem with time windows. We also considered two classes of heuristics for assigning 4-hour customers to time windows. Our implementation of our procedures were benchmarked for time and memory usage, as well as accuracy and error for the heuristics. We have shown that this method yields optimal solutions, or near optimal in the case of the heuristics, in a reasonable amount of time for Tesco.

		Heuristics		
	Optimal	Rule 1	Rule 2	Rule 3
Time (ms)	415.16	2.54	2.44	2.53
Memory (MiB)	767.0	1.0	1.0	1.0
Relative Speed	1	163	170	164
Relative Memory	1	1/767	1/767	1/767
Accuracy (%)	100	26	25	25
Relative Error (%)	0.0	2.4	2.7	2.5

Table 1—Benchmarks for three heuristics for four time windows with three customers per time window, and four 4-hour customers.

There are a few directions to extend this project. It would be of interest to use real-world data to compare the routes and objectives of our methods to Tesco's current implementation, and also how our results might change with real-world data instead of simulated data. Finally, developing a system to quickly and efficiently allocate customers' orders to specific delivery vans would be an essential step to building a complete logistics system.

Potential Impact

We have shown the algorithms we have developed can optimally solve vehicle routing problems like those faced by Tesco in only a few milliseconds. Our methods may help Tesco move away from a fully heuristic search to an optimal one—allowing more efficient delivery routes to be taken.

Una Benlic, Senior Data Scientist at Tesco, said:

> *The main aim of the project was to investigate whether there is a potential to improve on the existing Tesco systems that incorporate vehicle routing with delivery windows. At the moment, all such applications are based on heuristic search approaches, which provide an acceptable solution within reasonable computing efforts, but without a guarantee on the solution quality. This project provides an implementation of a Dynamic Programming approach that provides an optimal solution to the problem within milliseconds. Extensive evaluations of the performance in terms of memory and computing time requirements were conducted using synthetic data instances of different sizes. Given that the reported computing time is just a few milliseconds on average, the conclusion is that the developed approach could perhaps be considered as a component of a hybrid approach to the real-life problem of distribution to customers.*

Engineering and
Physical Sciences
Research Council