EPSRC
Engineering and Physical Sciences Research Council

InFoMM
Industrially Focused Mathematical Modelling

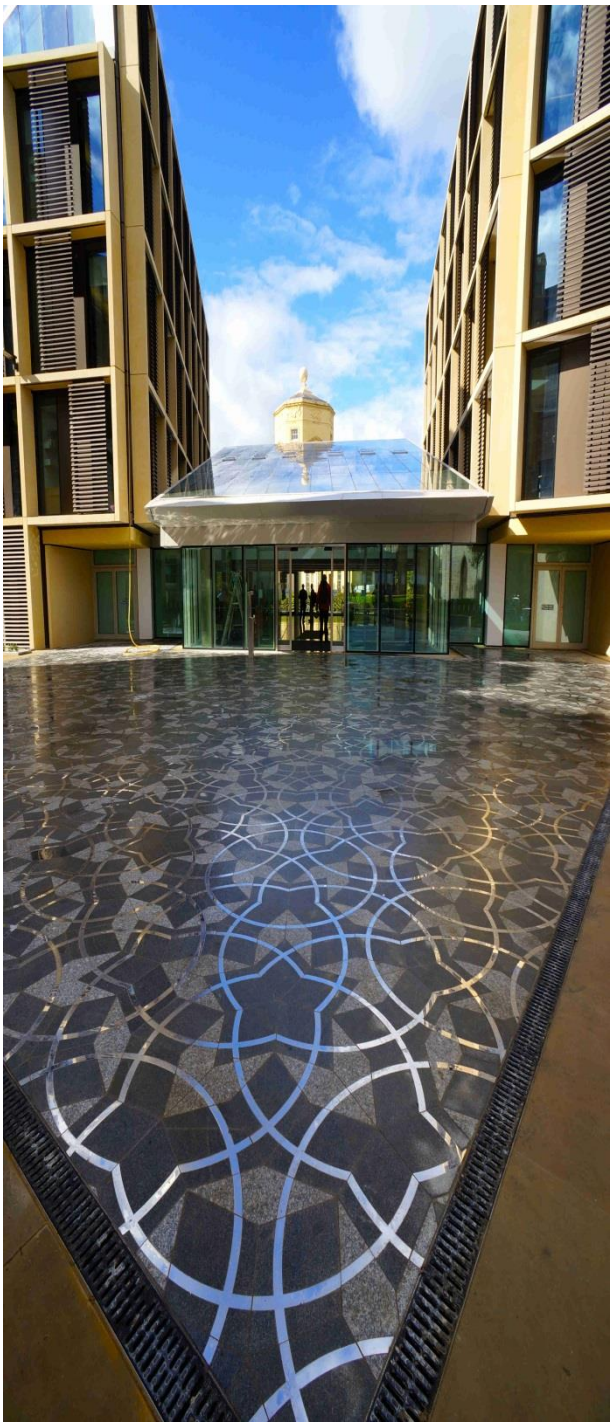
EPSRC Centre for Doctoral Training in Industrially Focused Mathematical Modelling

Robust Vehicle Routing

Jonathan Peters

UNIVERSITY OF OXFORD

TESCO



Contents

1. Introduction

Background

When someone orders a product online, there is a considerable amount of scheduling required for them to receive their order efficiently. Assuming the company that they ordered from has every product in their order waiting in a warehouse, remaining tasks still include:

- Grouping the order with others which need to be delivered to a similar location.
- Selecting a time window during which the customer is available to receive their order, which is compatible with the other deliveries to be made by the same van.
- Choosing the best route to take to ensure that the delivery is made within the customer's Time Windows.

We seek the route for a delivery van such that it delivers to each customer their order during the time frame when there are ready to receive it

The last of these problems is especially of interest to Tesco who wish to reduce the time their vans spend on the road in order to avoid unnecessary expense. However, they also must ensure that deliveries are made to their customers at the appropriate times in order to avoid angering customers, and potentially losing customers.

The main issue in reducing the time that their vans spend on the road is that travel times are uncertain. If we knew without doubt how long it would take for the van to drive from the depot to a customer, then this problem would be the known Travelling Salesmen Problem with Time Window constraints, in which case we could determine precisely which route the van should take and guarantee that all deliveries would be made on time.

An important facet is that the route needs to work even if there is unexpected traffic congestion

However, real-world factors such as unexpected traffic conditions complicate the problem. Even if we can predict with some certainty how long a journey will take, the possibility exists that traffic will suddenly congest, resulting in all remaining deliveries being late.

Therefore, Tesco is interested in how to choose the best route for the delivery van when the expected time required to travel along a specific road depends on the time of day (for example, taking into account traffic resulting from the end of the school day) and is uncertain and might take a different amount of time on different days. These different journey times occur due to factors which we cannot predict or control. In the context of uncertainty, we can no longer guarantee that a particular route would minimize the journey time, or that each customer will be delivered to within their respective time window.

In practice, when a Tesco delivery van sets out, it will usually contain deliveries for between 12 and 18 customers. Customer Time Windows start on an hour, and last for either one hour (for example from 2pm until 3pm) or 4 hours (from 1pm until 5pm). Usually up to 4 customers will share a common one-hour Time Window.

Our aim is to find a route which reduces the journey time while being *robust* to uncertainty.

Glossary of terms

- Route: A route refers to the order in which we visit a number of customers. For example, given the customers A, B, and C, there exist 2 routes which start and finish at A while only visiting B and C once: ABCA and ACBA.
- Time Window: Each customer gives Tesco a time interval during which they can receive their delivery. We refer to such time intervals (for example 8am – 9am) as Time Windows.
- Local Regret: The time between the **start** of a customer's Time Window and the time that the van actually arrives at that customer. Regret is the largest value of Local Regret over all customers.

- Scenario: We assume that traffic can be in 3 conditions within a particular time window: normal, semi-congested or congested. The sequence of conditions across all the Time Windows is called a Scenario.
- Robustness: Given a scenario, a route has a Regret. The robustness of a route is the maximum Regret over all scenarios. Robustness quantifies the performance of a route in the worst case situation.
- TSP: The Travelling Salesman Problem is the well-known problem in which we find the shortest journey a salesman can take starting at a fixed location, travelling to all customers, and returning to the starting point.

2. Dynamic solution to the TSP

Our objective is to find a route for the delivery van to take which departs the depot, delivers to all customers within their respective Time Windows, and returns to the depot at the end. We wish this route to be both robust, and to take as little time as possible. We assume that there are five Time Windows. While at first glance these Time Windows add an extra complication to the problem, they actually make it easier.

In Figure 1 we illustrate how the number of possible routes can be reduced as a consequence of the Time Windows. The complete graph on the left shows all roads that may be taken by a delivery van which starts at A, goes to B, C, D, and E returning to A at the end if time windows are not taken into account. However, given customer preferences, it is clear that , for example, we will never travel from B to D if B expects its delivery between 3pm and 4pm while D should be delivered to between 1pm and 2pm. By removing redundant edges, we reduce the size of the problem. The graph on the right is an example of what remains if we remove redundant edges. The only routes which remain as options are ADEBCA, and ADECBA. Previously, we would have to take into account all 24 routes. Thus, removing unnecessary edges simplifies this problem from 24 possible routes to only two. This procedure both makes the problem far easier for a computer to solve, but it also motivates a further simplification which is the entire basis of the approach which we take.

Time Windows allow us to reduce the number of possibilities we consider. This is because we must deliver to customers with early Time Windows before we delivery to customers with later Time Windows.

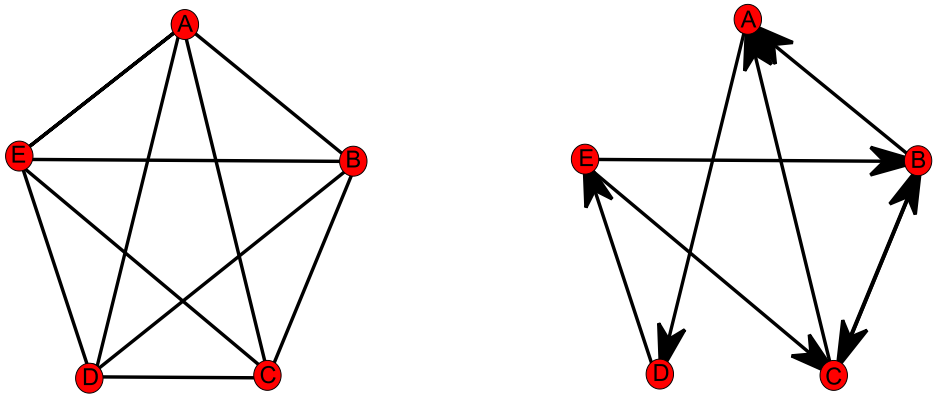


Figure 1: Schematic showing how taking Time Windows into account results in many edges being removed. All edges in the graph on the left are double sided, while the reduced graph on the left contains one-way edges.

We break the problem down into a series of smaller problems. Instead of looking at which order to visit all customers at once, we instead look at all of the customers which expect their delivery within one particular Time Window. This means that instead of looking at all the deliveries the van has to make during its journey, we only look at the number of customers within a particular Time Window, which is usually four. While there are many of these smaller problems, they are sufficiently easier to solve than the overall problem, and thus this approach is worthwhile.

We reduce a problem consisting of about 16 customers, into 56 smaller problems only with 3 customers each. These small problems are much easier to solve.

We solve a sub problem for every pair of customers in consecutive Time Windows. We fix a customer to be the first one we deliver to within the current Time Window, and a customer to be the first within the following Time Window. Then we compare all sub routes which visit all other customers within the current Time Window, and finish at the selected customer in the future Time Window. We record the most robust of these sub routes, taking into account how subsequent Time Windows are affected.

Once we have solved all of these sub problems, we are left with the problem stringing together these solutions, which we illustrate in Figure 2. By solving all of these sub problems, starting with the customers in the final Time Window and working backwards, we can iteratively determine what the best regret would be if we were to start at any given customer, and finish the journey from there.

We solve the problem in 2 steps. First we go backwards, computing the best case regret if we finish the route from a particular customer. Then we go forwards, following the smallest regret values to determine which route we should take.

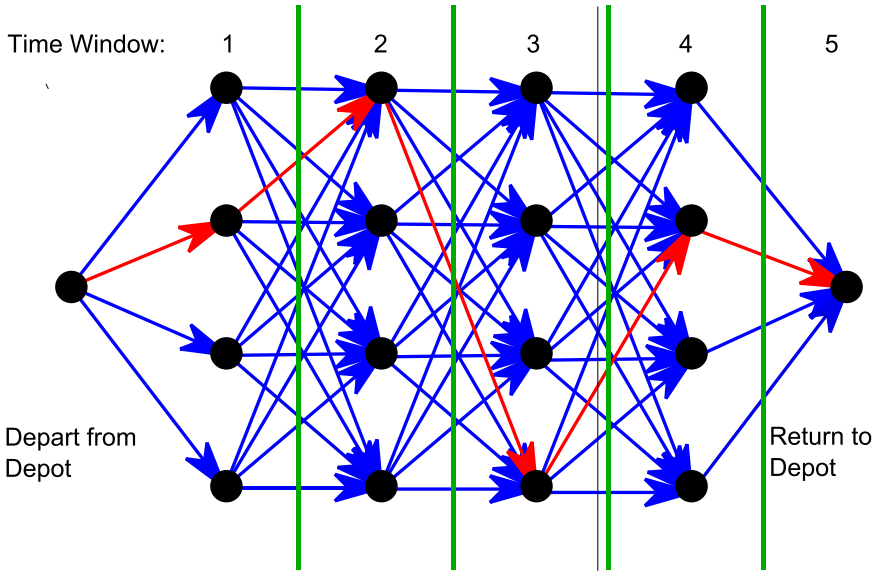


Figure 2: In this schematic, we show a problem with 16 customers which are divided across 5 time windows. Each black circle refers to the customer which we deliver to first within the corresponding time window. Each arrow is weighted with the robustness of the sub problem where we fix the start of the sub route to be the origin of the arrow, and fix the end of the sub route to be the target of the arrow. Finding the most robust route for the entire delivery is now equivalent to finding cheapest path from the left to right (for example, the route in red).

Once we have worked back to the start, we read off which customer in Time Window 1 is the best one to deliver to first. Next, we determine the best customer in Time Window 2 to deliver to first, given our starting point in Time Window 1, and so on. We determine in this way which customer we should arrive at first in each time window. The details of this optimal route are determined by revisiting the corresponding sub problems.

3. Finding a robust solution with recourse

In the previous section, we described the way that we might find a robust route which allows us to deliver to all customers regardless of the traffic conditions. However, can we do better? In reality, if the van drives along a road and observes intense traffic, then the traffic conditions are no longer uncertain.

A *recourse* solution is one that allows us to adjust our route as we observe the traffic conditions that are actually occurring.

This motivates the idea of a *recourse* solution: a route which we determine in advance in which there are different options depending on the traffic conditions that are actually observed. Better routes can be selected in this way. Once we observe that traffic is not intense, we can take the route which is best for low congestion.

In order to proceed, we must state more carefully what we mean by a Scenario. We assume that traffic can exist in one of three states: normal, partially congested, and congested. Note that when we say congestion here, we refer to traffic beyond that expected at a particular time in the day, i.e. we do not include standard rush hour congestion.

Next, we divide the delivery into the five Time Windows. We assume that the state of congestion only changes when moving from one Time Window to another. That is, within an hour block, traffic conditions remain fixed and only when moving to the next hour are conditions allowed to change. Finally, we assume that, if unexpected congestion exists, then during the time span of the delivery schedule the traffic will only get worse, i.e. if the current state of the roads is partially congested at 8:30am, we assume that at 9am it will either still be partially congested or will escalate to fully congested.

During each Time Window, the traffic takes a particular state. Therefore, if the delivery occurs over 5 Time Windows, then the traffic situation for the entire delivery, or scenario, refers to the sequence of 5 traffic states.

These assumptions give rise to the Scenario flow diagram which we show in Figure 3. The benefit of this approach is that there are only 11 distinct scenarios, and thus it is straightforward to carry out computations for all of them.

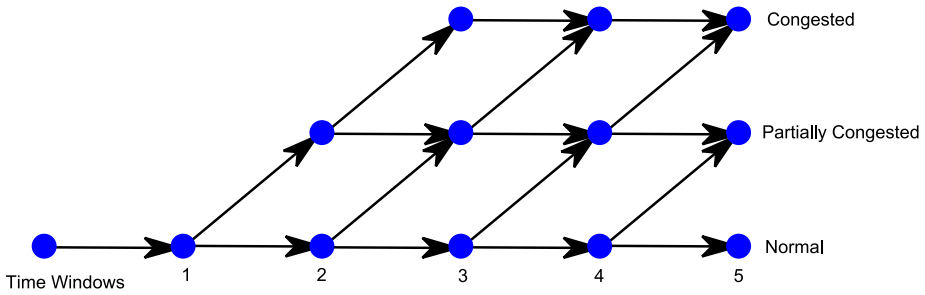


Figure 3: Schematic showing the way that unexpected traffic can develop during a delivery journey. The state of traffic at the start of the journey is always Normal. When transitioning to Time Window 2, traffic will either stay Normal, or else escalate to a state of Partially Congested. Each route between the node in Time Window 1 to any of the 3 nodes in Time Window 5 is a different Scenario which can actually happen over the course of a delivery. There are 11 such Scenarios.

Now we return to a *recourse* solution. First of all, we select a customer in the first Time Window that we go to first. Once we arrive there, we observe whether the traffic conditions are still normal, or have changed to a state of partially congested. For both of these possibilities, we plan a different route. Each of these routes will, in turn, branch off as the conditions that day are actually observed. By allowing the driver to adapt to the traffic he sees, we obtain a route which is both robust and efficient.

We write a *recourse* solution as a list of eleven different routes, one for each scenario. However, these routes must agree to some extent. If two scenarios are indistinguishable up to a particular time window, then we require that the *recourse* solution routes must agree up to that same Time Window. This means that the early parts of the journey need to be robust with respect to all eleven scenarios, while the last two hours only need to take one or two scenarios into account. The *recourse* solution is guaranteed to perform better than a single route chosen to maximise robustness, as a single route solution is merely a special case of a *recourse* solution.

We find this *recourse* solution algorithmically using a process similar to that described in the previous section.

Regret vs Journey Time

When computing the *recourse* solution, we find the route with the smallest regret value. We want to determine how this objective affects the total journey time, since reducing journey time is also of interest. The reason why we define local regret as how long after the **start** of the Time Window the van actually arrives is so that, by minimizing the maximum local regret, we are actively minimizing the overall journey time at the same time.

Minimizing regret is almost equivalent to reducing journey time but not quite. Our solution algorithm contains a parameter which allows to user to specify which is more important.

There is only one exception to this, a place where the objectives of minimizing travel time, and minimizing the maximum local regret, do not match. This exception is the last step of the journey, when we return to the Depot itself. Here we must ask ourselves, which would be better: to return to the depot at 4:30pm having delivered to the final customer at 4:10pm, which is 10 minutes late, or to deliver to the final customer at 3:55pm, but as a result return to the Depot at 4:50pm.

Instead of deciding which course of action is preferable, we create an artificial final Time Window for returning to the Depot. This is a parameter which is set by the user that has the effect of specifying which is more important: arriving back at the depot quickly, or robustly delivering to the customers in the penultimate Time Window.

Expanding the model

When designing the Dynamic Programming solution to this problem, it was convenient to assume that all Time Windows were exactly one hour in duration. In practice Tesco also have customers who have four-hour Time Windows, although there are fewer of these. There is a conceptually simple way to generalise the algorithm to this case, but it is inelegant and expensive to compute. To expand our model, it would be ideal to design a more natural way for these 4 hour Time Window customers to be included. Aside from this, we identify the following weaknesses of our approach which could be improved on:

- We assume that one set of scenarios affects the entire road network during a journey. In practice, if customers live in two different towns, we expect the traffic in those towns to be independent of each other. Accounting for this would require far more scenarios.
- We assume that, within a Time Window, the time to travel between a given pair of customers is constant.
- By minimizing the maximum local regret, we assume that the customers are well chosen so that no single customer is significantly out of the way. If this doesn't apply, then we might have one customer which is inevitably late, and thus dominates the attention of the objective function.

4. Results

The approach that is usually taken to solve the Travelling Salesman Problem is to construct a mixed integer programming formulation, and solve with appropriate software. We compare our dynamic programming algorithm to such a method, using problems which are as similar as possible given the different model assumptions. In Figure 5, we see that our dynamic approach performs approximately one hundred times faster for realistic problem sizes. Even when incorporating uncertainty, our approach still runs 10 times faster. Note that the mixed integer programming problem itself does not include uncertainty, and thus that it would be infeasible to approach the *robust* Travelling Salesman Problem this way.

We compare the speed of our dynamic programming approach to conventional mixed integer programming methods, and show ours to perform much faster.

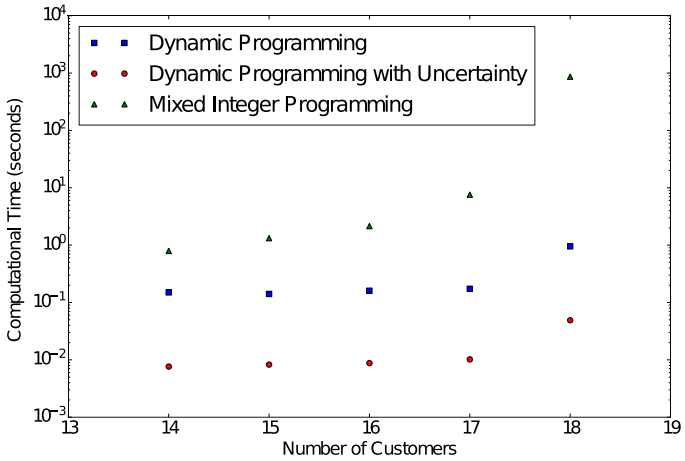


Figure 4: Graph showing the computational cost of our dynamic programming approach in comparison to conventional mixed integer programming methods.

5. Discussion, Conclusions & Recommendations

We have looked at how to incorporate uncertainty into a Travelling Salesman Problem and concluded that in order to include this feature, we need to make simplifications elsewhere. Thus we developed an algorithm which breaks the problem into smaller ones, making it computationally tractable, and is easily generalised to add uncertainty and *recourse* solutions.

A *recourse* solution has several advantages over a solution which is fixed in advance (an *a priori* solution). Given that a normal *a priori* solution is a special case of a *recourse* solution, it follows that a *recourse* solution will always perform at least as well as an *a priori* solution. In addition, having a solution which has been planned to include potential adjustments is more intuitive to a driver who encounters heavy traffic along a road. However, a *recourse* solution is a larger structure, and thus requires more computational effort to solve for.

We have presented a solution algorithm which demonstrates a considerable speedup over alternative methods. Thus we recommend that further attention be given to this approach, as it is fast enough to have additional complications included. Specifically, we suggest that the potential weaknesses mentioned in the previous section be given further scrutiny.

6. Potential Impact

If our assumptions concerning the time-dependence of travel times can be validated, then Tesco will be able to select routes for their vehicles which, even if they take slightly longer most of the time, would be able to handle rare extremities of traffic congestion, therefore avoiding the possibility of angering and thus losing customers.

Furthermore, the method we use of taking advantage of time windows to break the problem into sub-problems offers a new way of approaching this problem, which unlike conventional Mixed Integer Programming approaches, does not have trouble with the computational complexity due to uncertainty.

George Dikas, senior Data Scientist at Tesco, "*The outcomes of this mini-research project will be extremely useful for Tesco, since we now have a better definition for our problem, the notion of "robustness", a better understating of the computational time needed, and we have a first approach that we can elaborate on and build a business solution in the future.*

"I really enjoyed spending time with Jonathan and Raphael these past few weeks, and I am really satisfied about all the interesting insights gained and the overall outcome of this project.