Righting the Wrongs of Transport Optimisation

Professor Mark Wallace, October 2014

Executive Summary

Analytics provides a basis not only for understanding the past, but also for predicting the future – forthcoming customer demand, predictable resource bottlenecks and future cost spikes.

The step from understanding the past and predicting the future to **exploiting** that **knowledge** for better performance and lower costs **is called *optimisation***.

Optimisation is a black box to most people. Data is fed in, plans and schedules are written out, and either the results are ignored, or disruption follows. How can we have confidence that the software in the black box is doing the right thing?

To date there have been two types of optimisation. (A) The first type **automates** and accelerates the performance of human decision makers. It is a flexible approach which produces outcomes comparable with the human experts in a fraction of the time. However the solution is typically **far from optimal**. (B) The second type of optimisation uses **advanced algorithms** to produce highly optimised results. It does not emulate human decision-making process but implements logic and mathematics to tackle applications too large for the human brain. There is a **loss of flexibility** because algorithms have to be specifically tuned for different problem classes. The drawback of this type is its inability to handle special requirements.

Recent advances in optimisation techniques have enabled a third type (C) both flexible and optimising. The **Opturion** platform, developed by a group of world-renowned researchers in Australia, supports the third type for which the **flexibility and high quality** of its results have been proven at the coal face.

Righting the Wrongs of Transport Optimisation

Introduction

All software projects can go wrong for a variety of reasons. Transport and optimisation projects are particularly risky, because they are difficult to get right and errors lead to highly visible and costly outcomes, as seen with the Ariane 5 rocket crash in 1996 which resulted from a software overflow. Nevertheless transport optimisation projects are well worth implementing as they can bring about huge savings with a relatively small investment.

Transport planning and scheduling systems have been in active commercial use since their deployment for military logistics in World War II. To date there have been two types of system. The first type automates the behaviour of human dispatchers. This is a flexible approach, but its optimisation is quite basic. The second type incorporates optimisation algorithms tuned to a specific version of the transport problem. The drawback of this type is its inability to handle special requirements.

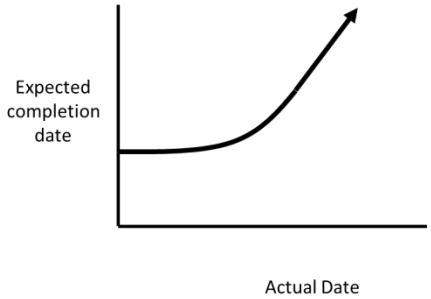
Recent advances in optimisation techniques have enabled novel transport planning and scheduling systems to be implemented, that are both flexible and optimising. The Opturion platform, developed by a group of world-renowned researchers in Australia, supports a number of successful commercial transport optimisation systems of a third type for which the flexibility and high quality of its results have been proven at the coal face.

Software Risks

Managing software projects can be problematic because only once the software has been implemented is its functionality precisely specified. It is easy to stipulate that "vehicles should take the best route" or that "drivers should be allowed sufficient time for breaks" for example, but such requirements are imprecise. Moreover, it is easy to omit specific mention of obvious constraints – that a B-double truck cannot enter a driveway, for example. When the software is finally tested it often proves not to have met the actual customer requirements. Furthermore, by the time the software is completed, these specifications may have changed; for instance, there may be new laws on driver fatigue management, new routing methods and even new conditions at customer sites. Three months is a long time in business!

Novel software often enables, or imposes, novel ways of working. The opportunity for improved efficiency can be experienced instead as a threat by the users of the software, and as a result the new process may be undermined by the very people who need to make it work.

Finally, while it is possible to observe progress during the construction of a house, the construction of software is practically invisible to everyone except the implementers. The mantra that a software implementation is "90% finished" can be repeated for months! As software projects go on, their expected completion dates often start to slip, faster and faster until the time till completion ceases to narrow. Under these circumstances the project will never end!



Transport Optimisation Software (TOS) Risks

As the images above illustrate, transport disruptions resulting from software bugs are obvious and expensive.

Optimisation adds another layer of difficulty to the usual software risks. If the software fails to meet all the user requirements, or when the requirements change over time, adapting optimisation software is challenging and time consuming. This is because the algorithms encoded in the software usually need amending in order to find good solutions satisfying the new requirements. Developing a new algorithm may cause old requirements to be violated, or may need more computing resources, may yield lower quality solutions, or may even fail to scale to the original problem size.

A fundamental issue when managing a software optimisation project is how to tell whether a solution is "optimised". When a problem has previously been tackled by hand, the solution produced by the software may be radically different, making it very hard for the problem experts to determine whether the new solution is either correct or of high quality. Worse still, even if the new solution proves to be correct and looks good to the human expert, there may be much better solutions undetected by the software.

For academic problems, such as the "Travelling Salesman Problem" (of which more will be discussed later), it is sometimes possible for the algorithm to prove that the solution it has found is indeed the best possible solution. Unfortunately, even the most powerful computer grids are unable to complete such proofs for real-world application problems. Moreover, from a theoretical point of view, this limitation will never be overcome by bigger and faster computers.

Why bother with TOS?

Given the risks of Transport Optimisation Software, it is tempting to ask why one should bother with it at all! The answer is that when it works, TOS projects yield tremendous benefits which dramatically enhance the competitiveness of the companies where they are deployed. British Telecom, for example, used optimisation software to reduce the travel and waiting times for their field engineers. As a result, the average number of jobs per day for each engineer increased from 7 to 8, yielding a saving of 150 million pounds per year.

Another example was the use of optimisation by the RAC to reduce advised response times (ART). This is the average time required for a patrol to reach a breakdown which is advised to customers awaiting rescue. By reducing their ART below that of their competitors the RAC quickly increased their market share and secured their position in an increasingly tight UK market.

Some companies that have failed to take advantage of optimisation technology have found themselves unable to compete, and ultimately closed their doors.

Automated TOS

Automated TOS Components

Assuming an organisation is committed to installing optimisation software for its logistics and transport, there are various types of TOS available.

The first type comprises four components:

- Communications
  - real time communication with vehicle drivers and customers
- Command and control
  - a command centre showing the current state – which may include vehicle locations if the real-time information is available
- Recording
  - delivery notes
  - driver working hours
  - contractor information
- Decision support

Automated TOS Decision Support

In this article, we focus on the fourth component, decision support. The aim of automated TOS is to automate processes previously done by hand, including telephoning, moving markers on maps and so on. Successful TOS do what people do, only faster and more consistently.

Consider what controllers do when allocating vehicles to service customer requests. Requests are handled one at a time and for each request the controller considers which vehicle is best placed (in both time and location) to service the request. If the request is a delivery, for example, the controller

might consider the route currently planned for each vehicle, and where in each route the request would most economically be served – respecting constraints such as the weight and volume of goods, vehicle size, customer time-window etc. The request is then assigned to the vehicle and position in its route that is both feasible and most cost-effective.

Once one request has been dealt with, the controller considers the next request, until all requests have either been assigned to a vehicle or – if a request can't be serviced by any vehicle in a way that satisfies the constraints – rejected. A rejected request may be postponed to another day, delegated to a contractor, or turned down completely.

Automated TOS does the same thing, only faster, and guaranteeing the assignment of each request is cost-optimal.

The benefit of this TOS is flexibility. For each potential assignment of a request to a vehicle, all the user requirements can be checked, even if they are application-specific. Examples of such requirements are:

- Empty pallets should be collected from customers at their next delivery
- Incompatible freight should be separated
- Time windows
- Big loads should not be carried uphill

However, there are other constraints – ones which cover multiple requests and/or multiple vehicles, such as a requirement to balance the work assigned across a vehicle fleet – that cannot be handled on a per-request basis.

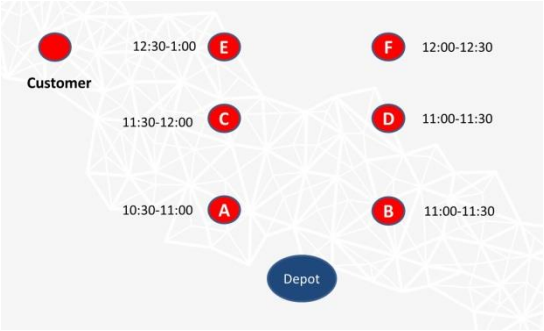
Automated TOS Performance

Automated TOS systems add requests one at a time. Adding one request is efficient for a computer which can quickly check each possible way to add a pick-up and delivery to an existing vehicle route. Let's say that there are 10 locations on the current route. The number of way to insert a pick-up and a delivery into the route is at most 100 (for each of the 10 places on the route to insert the pick-up, there are at most 10 possible places to insert the delivery)[1]. On the other hand, if there are 50 vehicles, the total number of alternatives is 100 per vehicle, which is 5,000 possibilities altogether. Choosing the best is easy for a computer, and even possible for an experienced dispatcher by hand.

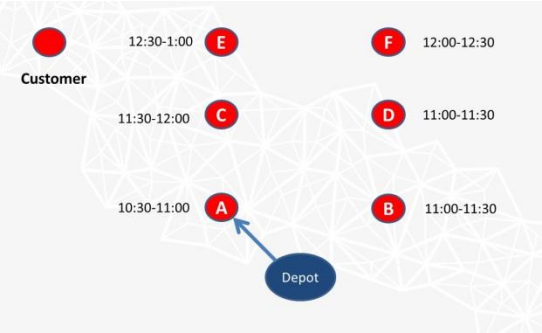
How good is the resulting schedule? Would it be possible to do better than this?

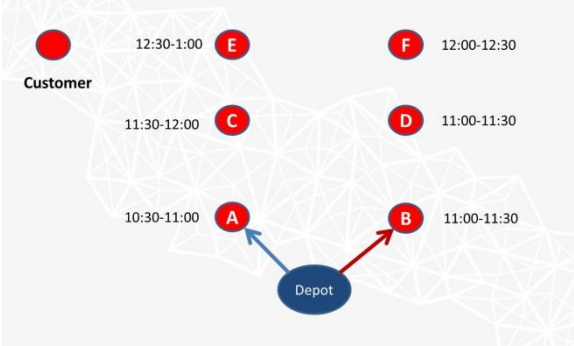
1. For mathematical readers, a tighter bound is (10 + 9 + … + 3 + 2 + 1)

Let us take a simple example to illustrate why allocating requests one-at-a-time does not work very well. Suppose we have two vehicles and 6 deliveries to make. The following diagram shows the depot, the 6 customer locations and a time-window for each location.

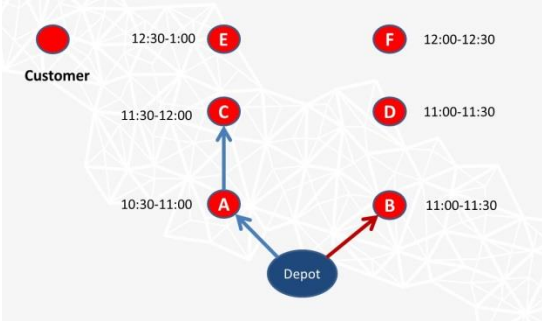


The requests are assigned in the order A, B, C, D, E and finally F. Let's assign request A to the first vehicle. Now for request B, the nearest vehicle is the one still at the depot, so we assign that one.

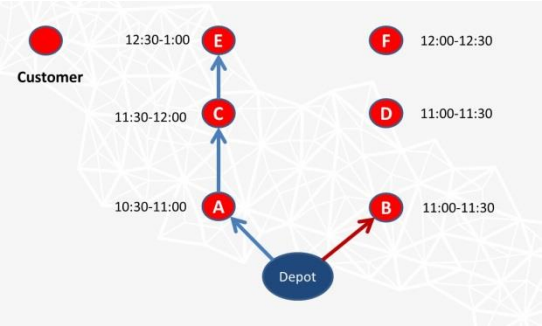


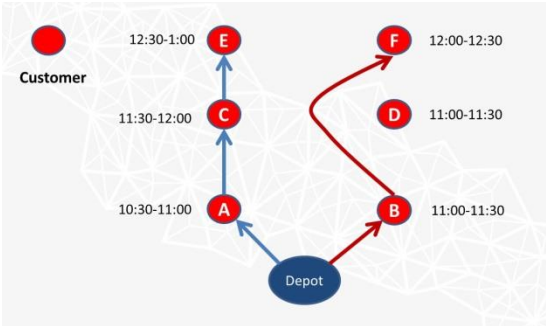


Request C is assigned to the blue vehicle.

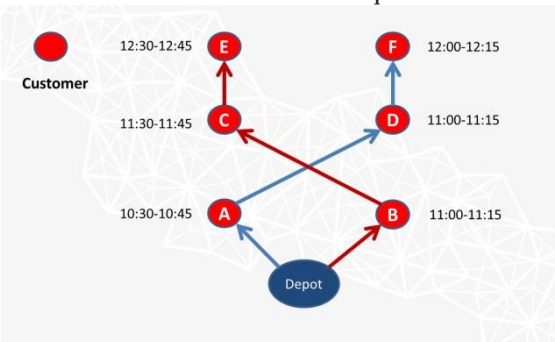


Because of the time window on request D, it can't be assigned to the red vehicle, and it cannot be slotted into the blue vehicle between requests A and C. The last two requests are allocated easily:





The final schedule misses out request D, which is postponed to another day, or cancelled. However there is a perfectly good schedule which handles all the requests as follows:

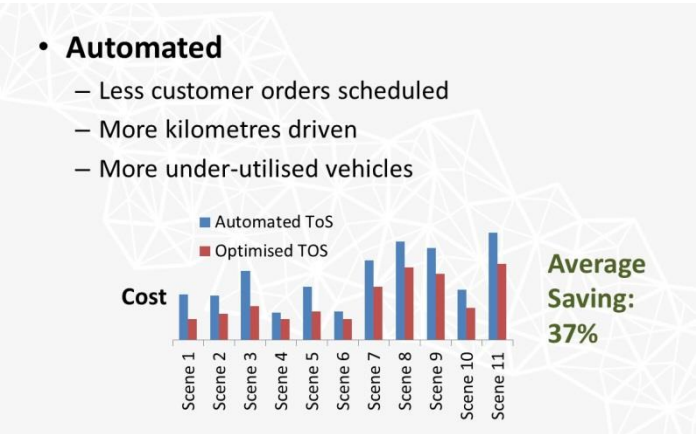


Unfortunately, when scheduling one request at a time, this optimal solution is not found.

Naturally this example could easily be scheduled optimally by hand. However it illustrates how a computer algorithm, handling requests one-at-a-time, and making the optimal choice for each request, can produce a non-optimal solution.

We took a set of examples from real life and compared the schedule produced as above with the optimal schedule computed by Opturion's software. (For these tests we even improved the one-at-a-time algorithm so that the next request allocated was chosen intelligently.) The cost of a solution was a combination of the number of vehicles used, distance travelled and penalty for unserved requests.

The results showed a substantial difference between the solution obtained by the Automated TOS algorithm, and the optimal solution computed by the Opturion software.



This diagram illustrates that the total cost of the optimal schedule was nearly 40% less than the cost of the Automated TOS schedule!

Cost Minimising TOS

Computation time for Automated TOS versus Cost Minimising TOS

Naturally the cost gains through optimisation make it well worthwhile to apply more sophisticated computer algorithms to get better solutions. However, finding an optimal solution to a commercial-scale problem is extraordinarily hard. For the Automated TOS algorithm that handles one request at a time, we noted that the computer had to choose the best among 5,000 alternatives for each request. To find the optimal solution for such a problem (with 50 vehicles and 500 requests), we have to

consider all ways of allocating all the requests to all the vehicles in combination. The number of alternatives is unbelievably large. Scientists estimate there are about 10^{80} atoms in the universe. Amazingly the number of alternative ways of scheduling 500 requests among 50 vehicles is greater than the number of atoms in the universe!

Because Automated TOS considers such a relatively small number of alternatives, software using this approach can produce solutions very quickly. Commercial software that solves vehicle routing applications in "real time" – that is, a few seconds – can be categorised as relying on Automated TOS algorithms. The benefits of speed of response must be weighed against the potential additional cost of the solution generated.

Algorithms for Cost Minimising TOS

There is no computer that can check 10^{80} alternatives and select the best. Indeed a million parallel machines, each machine checking a million alternatives per second will still take around 10^{60} years to find the best one. That is more than the life of the universe!

Cost minimising TOS uses sophisticated mathematical algorithms to find high quality solutions by exploiting the structure of the problem, rather than naively searching through 10^{80} alternatives. Different transport optimisation problems need different algorithms.

The simplest transport optimisation problem is to find the shortest route, starting at the depot, visiting a given set of locations, and returning again to the depot. The problem is often known as the "Travelling Salesman Problem" – TSP. Mathematicians have worked on algorithms for solving the TSP since the 1800's. Larger and larger instances have been solved since 1954, when a problem with 49 locations was solved to optimality. Recently TSPs with over 20,000 locations have been solved – and "solving" means proving that the solution found is the best among all alternative routes, even though there are more such routes than atoms in the universe.

The next class of problem is the vehicle routing problem (VRP) which is similar to the TSP, except that there are multiple vehicles. In the capacitated VRP, each vehicle has a certain capacity and a varied given quantity that is to be delivered at each location. This is surprisingly different from the TSP, and optimum solutions can only be found – and proven optimal – for problems with a few hundred locations.

When requests involve moving freight from one location to another, the problem class is called Pick-up and Delivery (PDP), which requires another class of algorithm. Both the VRP and the PDP problem can be extended by adding time-windows, leading to two new classes of problems.

Just as all these different academic problem classes require different algorithms, different commercial freight transport problems are currently solved by different optimisation algorithms. Example problem classes involve, in addition to requests with loads and time windows, and vehicles with costs and capacities:

- Drivers with shifts, overtime and required driver breaks
- Vehicle types and freight types with compatibility constraints
- Multi-objective problems with a requirement for fair distribution of work across different classes of drivers
- "Arc" routing, where all houses on a street must be visited (for newspapers, refuse collection etc.)

Inflexibility of Cost Minimising TOS

If each problem class uses a different algorithm, it is also a different product. Surprisingly, perhaps, products cannot easily be combined. Though a supplier may offer one product to handle driver shifts, and another product for arc routing, the supplier might not be able to support a product which handles both.

Moreover, the requirements for freight transport are always changing. New fatigue regulations dictate different requirements on driving hours, number and duration of breaks and minimal off-duty time between shifts. A product and algorithm designed to handle driver breaks is typically not designed to handle new fatigue requirements, and therefore cannot be used when the law changes.

Because each organisation is different, each freight transport application throws up novel constraints. Practical examples are that:

- When collecting freight from multiple locations, heavy loads should not be carried uphill
- When freight is delivered and also empty pallets are collected, delivery and collection should be done on the same visit, to minimise disruption at the customer site
- When (certain) people are transported, male and females should not be carried in the same load

Since mature commercial cost minimising TOS uses algorithms which work only on the problem classes for which they were designed, many current users have to configure the software to overcome its limitations. For example, to enforce the constraint that heavy loads should not be carried uphill, the software might be configured to ensure that request A, which is located at the top of the hill, should always precede requests B, C and D on any route. However, on certain days one of the other requests may actually involve a light load, and the route generated by the software, still enforcing the precedence constraint, may be much longer than necessary. Moreover , the shortest route from B to C may actually go past the location of A, but the algorithm preventing request A from being served until after C, will not schedule the collection at A until the vehicle has travelled all the way to C and back!

The consequence of cost minimising TOS is, in short, inflexibility to handle new requirements and software configurations that often prevent cost minimisation from being achieved.

New Generation TOS

Optimisation state-of-the-art

Over many years the performance of computer hardware has doubled every three years, so that the computing power in a mobile phone is far greater than the world's fastest (liquid-nitrogen cooled) computer in the 1980s.

Amazingly the same kind of performance improvement is now being achieved in optimisation software. Software tasks which required an hour of computation a decade ago, take a second today – on the same hardware. The result is that recent optimisation algorithms can achieve both optimisation and flexibility at the same time.

Flexibility does not mean that a single algorithm can handle all requirements that could ever arise in the future. The key benefit of flexibility is that when the algorithm is extended to handle a new requirement, it still includes the original algorithm. Thus, another extension can be added to handle another new constraint by building on the extended algorithm rather than devising a new one. The

result is that a new generation TOS solution has the capability of supporting any subset of old and new requirements that the customer demands.

Consider a set of requirements, like the following:

A. Big loads should not be carried uphill
B. Incompatible freight should be separated
C. Load packing and route planning should be integrated so that loads can be accessed in the optimal route sequence
D. Empty pallets should be collected from customers at their next delivery
E. Queues at loading bays should be precluded by integrated routing and scheduling
F. Contractor loads should meet a certain minimum value
G. Cross-docking should be optimised

If seven specialised cost minimising algorithms were developed, one for each requirement, then a customer with a combination of requirements would still have no solution. Additional algorithms would have to be designed for each combination; A+B, A+C, … F+G. There are a surprisingly large number of combinations of two or more from the set {A,B,C,D,E,F,G}. To be precise there are 128 such combinations.

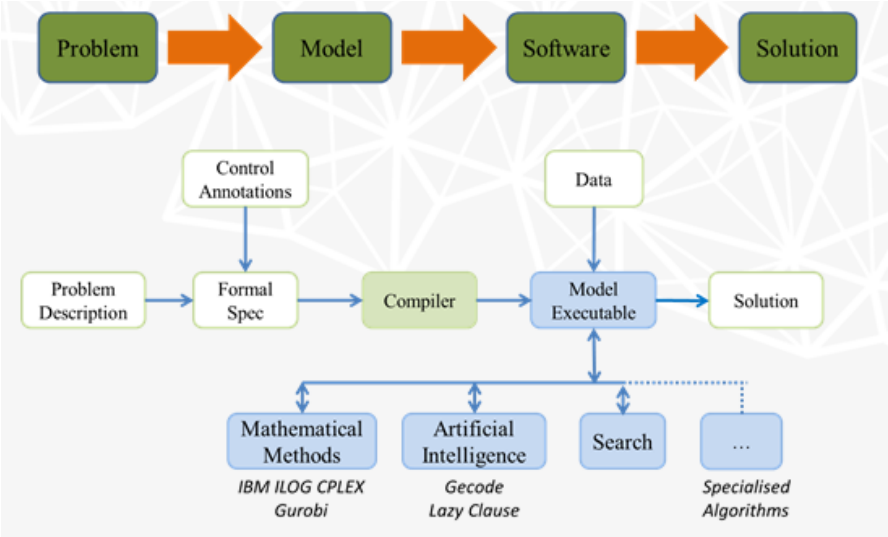
New Generation Algorithms

A new generation of TOS algorithms have recently emerged from research laboratories. These algorithms optimise far better than the cost minimising TOS algorithms because they integrate ideas from several areas of optimisation. The game-changing aspect of this new generation is, however, in their ability to handle ad hoc constraints and requirements.

Any new requirement can be added and it will be automatically enforced by the algorithm. Constraint "propagation" techniques are used to ensure that, as improved solutions are constructed, choices which would eventually lead to a requirement being violated are immediately excluded. The result is that feasible, high quality solutions can be quickly found from amongst literally thousands of billions of alternatives. Using a new generation TOS, the extensions to handle each of the requirements A,…,G are all compatible, and thus any combination is available from the extended solution.

Opturion Optimisation Platform

The Opturion platform integrates optimisation techniques from mathematics, operations research, physics and artificial intelligence. It is the first commercially available platform with the unique ability to map a single model down to such a broad combination of optimisation techniques. The platform enables users to write their problems down in a clear precise and unambiguous way as a "model". The model is compiled onto different optimisation techniques chosen to match the problem, and the resulting program is run whenever required (every minute/ hour/ day/ week/ month) against the current data. The platform architecture is shown below.



The platform has been in commercial use for two years and is used to schedule freight delivery onto fleets of trucks for several Australian companies. It is also used for procurement and shipping planning, maintenance scheduling, project planning, rostering and timetabling.

****

Author:
Mark Wallace, mwallace@opturion.com
Opturion, www.opturion.com

Professor Mark Wallace leads the Optimisation group at the Monash Faculty of Information Technology, and is CEO of Opturion, a company spun out in 2011 from the National ICT Australia centre of excellence to commercialise their world-leading optimisation platform.
He is a world leader in hybrid algorithms, known for combining academic contributions to the field, with solving real world problems. The software developed by his team at Imperial College London, in use at over 700 research institutions, was bought by Cisco Systems in 2004. His highly-cited book describing the system was published in 2007.
His transport optimisation research covers airlines, and train scheduling as well as road freight logistics and public transport. His algorithms have been deployed in organisations including British Airways, Qantas, the RAC, Hong-Kong's MTR and Wincanton Transport.