Opturion

Effective Optimisation

Oct 2015

Professor Mark Wallace

Introduction

The massive growth in mobile computing and online connectivity is driving a revolution in business processes. Business transformation occurs when this data is used to forecast demand and optimise resources. Standardly optimisation dramatically improves customer service, minimises the impacts of disruption, and saves from 10% to 30% on resource costs. Gartner predicts that "by 2018, optimization will no longer be a niche discipline; it will become a best practice for leading organizations to address a wide range of complex business decisions".

Optimisation technology is also undergoing a revolution. Today's optimisation algorithms are thousands of times faster than those developed in the early 2000's, and todays solutions are thousands of times more flexible and scalable than commercial products from that time.

To extract the benefits of optimisation, it is important to reap the benefits of the technological advantages and not be seduced by user interfaces fronting out-of-date technology.

Impact and Benefits of Optimisation

Organisations typically seek help with their planning and scheduling processes when they undergo a change. Often this is due to growth, when they find their current planning task is becoming unmanageable. Sometimes the process depends on a few key employees, who leave the company.

Optimisation software can plan much better than even the most experienced human schedulers, and the system automatically handles the increased complexity as the organisation grows. The immediate consequence is that the schedule is guaranteed to be objective and fair. It can respond to changes in requirements: such as shortening customer waiting times, subject to a given cost bound; or by contrast fixing the waiting time limit and minimising cost. The new process is repeatable, and cannot be impacted by the absence of a crucial staff member.

The surprising outcome is how much cost saving is typically achievable by the software. Savings of 30% over scheduling by hand are normal – a result that is often shocking to the "experts" who previously created the schedules. Replacing older optimisation software also brings serious benefits. Usually this is because the older software cannot handle some new business requirements, but even in case the requirements and problem size can be handled by the previous software, savings of over 10% are typical.

The third benefit is a dramatic improvement for all stakeholders, including customers, employees, contractors and shareholders. The customers find that less orders are delayed; the staff find that last-minute crises, and consequent placating of angry customers, are dramatically reduced; and shareholders see more satisfied customers as well as an improvement in the bottom line.

The Challenges of Optimisation

Assigning resources to tasks seems easy. However sometimes assigning the "best" resource to the first task, makes it very difficult, or expensive, to assign a resource to a later task. The trouble is that to get the best result it is necessary to consider all the tasks and all the resources at the same time. This turns out to be surprisingly hard.

Suppose there are 5 resources and 10 tasks to be covered. One (not very good option) is to assign all the tasks to the same resource. Another is to assign 2 tasks to each resource, but this may not be the best if there's one very long task and 9 quick ones. If we want a computer to check all the options and choose the best then the number of options is almost 10 million. It can do that pretty fast.

However if the number of tasks increases to 20 and the number of resources to 10, then the total number of options to check is: 100,000,000,000,000,000,000. This ridiculously large number is more than the number of atoms in the universe and would take 1000's of years to check, however big or parallel your computer.

Such assignments are made every day in organisations all over the world. In almost every case the assignment chosen is not the optimal one. It can't be, because there's no time to check all the options.

In the appendix we summarise a few methods used currently to optimise problems like these.

Cost and Delays in Optimisation Projects

Getting the Requirements Right

The first challenge in any software project is to specify the requirement. Requirements often turn out to be either imprecise or even wrong. Requirements suffer from known unknowns, but during the project it is the unknown unknowns that really blow things off course!

As a simple example, a requirement that states "set the weekly production so it matches the demand" sounds simple. However this requirement probably omits important issues: additional production may be needed for walk-in clients; some production may need postponing due to a lack of raw materials; the demand may only be revealed as the week progresses; and so on.

We scheduled a season's flights for a major airline, and were shocked to find that it was impossible with their current fleet! The airline assumed our software was at fault, until we finally realised their current schedule violated their own specification of the requirements. In the end our optimiser improved their schedule by minimising the number of violated requirements, which actually improved their punctuality.

Getting the Optimiser Right

The second challenge, for optimisation projects, is implementing the optimiser. Commercial Off The Shelf (COTS) software is often proposed as a good starting point. Unfortunately adding new features

to a COTS optimisation solution typically takes longer, and yields a worse outcome, than starting from scratch. An advantage of starting from scratch is that it avoids the inefficiency and inflexibility of old algorithms. Optimisation algorithms have been getting a ten-fold improvement every three years so old algorithms are now hopelessly uncompetitive.

However almost all optimisation products, and optimisation consultants, are wedded to a single optimisation technology (either mathematical programming or meta-heuristics or rule-based search and inference). Predicting the best technology for an application is impossible – the requirements change and any change in requirements can result in a different technology working better. Choosing the wrong technology can increase the time needed to implement an optimisation solution from a few weeks to a few years! Worse still, the solutions returned by the wrong technology, even after all that implementation, will be much more costly in resources and of a lower quality for the customer.

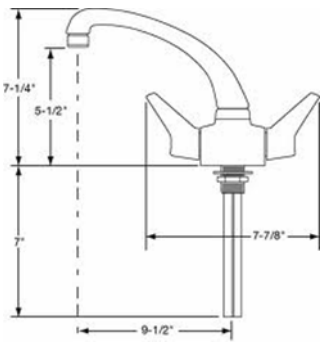
Methodology for Effective Optimisation

The time and cost involved in delivering an optimisation solution is mostly spent on getting the specification right and deploying the right optimisation technology. In fact most projects fail on both counts! The time taken to deliver a tailored commercial solution is typically measured in years, and by the time a solution has been delivered both the business and the technology has moved on.

A methodology for effective optimisation must minimise both

- the time and effort required to get the specification right
- the time and effort required to use the right technology to implement the solution

Get the specification right



The methodology required must be Agile. The initial requirements must be quickly tested by implementing a solution that delivers results for the customer to analyse. The customer almost always identifies shortcomings in the solution, and the requirements are augmented, or changed, to avoid the shortcomings. The new requirements must be quickly implemented so as to repeat the cycle repeatedly until all shortcomings have been eliminated. This iteration enables the correct specification to be finally discovered.

Deploy the right technology

Implementing a software module that returns the best solution to a given specification is not simple. There have been hundreds of years of research on standard problems, such as those discussed in Appendix II. Each variation on the standard problem generates years of research. Efficient deployment depends on postponing the choice of technology until all the requirements are agreed, or being able to quickly change the choice of

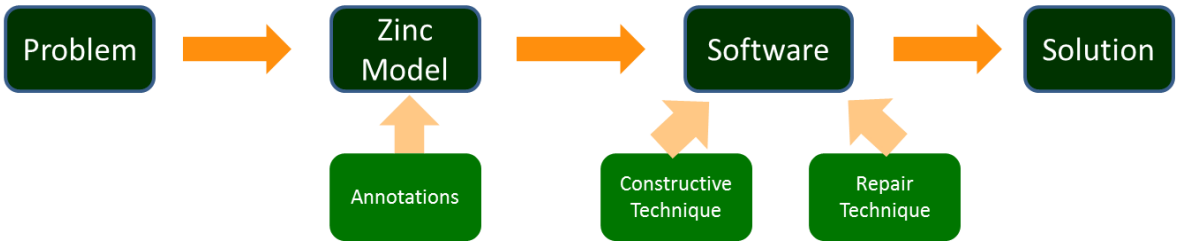
technology as the application requirements emerge. The best solution is achieved by combining the best technologies into a single application-specific approach.

The Opturion Optimisation Platform

The Opturion platform supports our Methodology for Efficient Implementation.

For getting the specification right, Opturion offers a runnable specification language, called ***Zinc.*** The specification written in the Zinc language is tested by running it on application data. Although application users often have difficulty expressing their requirements, they quickly spot errors in the solution produced by running the specification. This provides a basis for updating the specification, often many times, before the output looks correct.

Running on a small data set is useful, but many of the real issues come up with a full data set – for example running on the data for a whole day, month or year depending on the application. To test a large data set it is necessary to use a technology that can scale appropriately.



The Opturion platform includes a compiler that maps the specification to a program that can be run against the data. This compiler is controlled by adding ***annotations*** to the Zinc model. The annotations tell the compiler which technology to deploy for which parts of the specification. When the specification changes, or when a radically different data set is used, then the compilation with the current annotations, may not scale enough to return good solutions within acceptable timescales. When this happens, there is no need to redesign the application or re-implement the solution. In the Opturion platform it is only necessary to update the annotations, so as to deploy a more appropriate combination of technologies. This can be done in a few days.

Opturion Product Status

The Opturion platform is the result of some 60 person-years of research and development, by a group of world-leading researchers. The productisation of the platform started when Opturion was founded in 2011. Over the next year the platform was embedded in the SmartTrans fleet management solution, replacing a number of optimisation engines that were previously used to support different SmartTrans customers. The Opturion optimiser embedded in the SmartTrans e-Solution went live in 2012, and has been in commercial operation for SmartTrans ever since.

The Opturion platform maps high-level Zinc models down to a number of different technologies, and over the years 2011-2015 the integration with different technologies and the compiler itself have been extended to handle large scale applications (with 10,000 variables and constraints), better algorithm control (enhanced learning and search for our Eureka prize-winning “CPX” solver), and embeddability (web, spreadsheet and database interfaces).

The platform now support live commercial applications across multiple application areas. For a Fortune 500 customer, the Opturion platform has replaced the world's most widely used production scheduling software which did not scale to their stringent production scheduling requirements. For a Royal Australian college, the Opturion platform allocates participants in complex nationwide examination events to times and locations subject to constraint on compatibility, experience and availability. For mobile workforces, the Opturion platform dynamically optimises the allocation of tasks to engineers to meet deadlines and maximise the engineers' productivity.

The Opturion platform now solves strategic infrastructure planning problems, tactical resource allocation problems with complex compatibility and time constraints, and finally it solves dynamic operational optimisation problems. Savings reported by our customers include $200,000 per month for a transport application, reduction of customer service times of 3 hours to 15 minutes for an online mobile workforce application, and elimination of a massive re-rostering problem costing hours of effort per day as well as serious disruption and manpower costs.

Appendix I

Optimisation Technologies

Optimisation software dates back to the 1950's when operations researchers tackled the requirements of military supplies. The algorithms exploited at that time have been extended and enhanced over the years, but the core technique of reducing the requirements to a simplified numerical form remains the same.

These mathematical algorithms have been applied with great success to a number of standard benchmark problems, such as the "Travelling Salesman" problem. The distance travelled by the salesman can be minimised for longer and longer tours, visiting more and more locations.

Constructive Methods

These methods make a sequence of choices – for example which location is visited first by the travelling salesman, which is visited second, and so on. As the algorithm progresses the system finds better solutions, and also finds better "lower bounds". For each lower bound, the system proves that there is no tour with a distance less than that.

The advantage of constructive methods is the known gap between the best solution and the lower bound. Moreover there are constructive methods that scale to a huge size on standard benchmarks (the optimal route for a Travelling Salesman tour of 10,000 locations has been found and proven optimal).

Unfortunately the simplification of requirements to numerical form can be awkward, and in many cases it makes the problem too hard to solve for constructive methods. For example by adding "side-constraints" (such as extending the Travelling Salesman problem to include picking up and delivering goods, with a set of vehicles of limited capacity) the number of locations that can be solved to optimality reduces to about 100. Adding further side constraints the problem can become too large to load into the optimisation software module.

Repair Methods

These methods start with a simple solution which may be very poor – for example the travelling salesman visiting all the locations in some fixed order (e.g. alphabetical). Another solution is then constructed by "repairing" the previous solution, by making a small change, such as switching the order of two locations.

The advantage of repair methods is their scalability, and flexibility. For problems with many "side-constraints" that break constructive methods, repair methods can quickly produce solutions that are far better than any found by an expert.

The main disadvantage is that solutions don't necessarily satisfy all the side-constraints. Repair methods are driven to satisfy constraints by imposing a penalty on each violated constraint. However setting the right penalties all the constraints is an art - and if the penalties are not quite right the solutions produced may all be unusable.

Another disadvantage is that the solutions may be far from optimal – there is no way of knowing how good a solution is. A different repair method might therefore produce much better solutions for the same problem.

Inference Methods

With roots in Artificial Intelligence, inference methods allow the problem to be modelled accurately, including the awkward side-constraints. In principle inference methods ensure that all solutions found satisfy all the problem constraints. Unfortunately inference alone does not suffice to solve real-life problems and so the inference has to be embedded in a constructive or repair algorithm. For some models this combination works well. For other models the computational cost of inference is excessive, or (on the other hand) so little inference is possible that a complete candidate solution must be constructed before any useful inferences are made. In either case the computational overhead of inference slows down the algorithm unacceptably.

Commercial Offerings

Optimisation technology has improved dramatically in recent years, with tests revealing an exponential rate of improvement similar to that achieved in hardware until recently.

Commercial solutions that have already achieved market penetration do not benefit from these recent advances. Typically the companies who sell them are not their original developers, and so they are unable to replace the core of the system with more recent scalable, flexible and efficient algorithms.

Optimisation consultants who build new algorithms from scratch typically have experience with a single optimisation technique and may therefore fail deploy the method, or combination of methods, best suited to the problem at hand.

From the customer standpoint, it is very difficult to choose the best supplier or product: only after a solution has been implemented and tested on a month's worth of data can its quality be assessed. Comparing two suppliers or products requires both to be implemented and tested as above. Since this takes too much time and effort to be used as a purchasing process, the only basis for comparison is by reference to previous customers who may have different motivations for making a recommendation.

Appendix II

Mathematicians and operations researchers spend years studying different classes of optimisation problem. Classic problem classes includes the "Travelling Salesman Problem" (TSP) the "Job-Shop Scheduling Problem", the "Bin-Packing Problem" and etc. If your problem is one of these, then you can find an efficient scalable algorithm on the web to meet your requirement.

The surprising fact is that if your problem is at all different from one of the standard classes, then the algorithm you need is likely to be something completely different. For example adding "time windows" in the travelling salesman problem – a (different) restricted time-interval during which each (different) location should be visited – creates a new problem class: the TSPTW solved by different algorithms. Now require that a certain weight of goods must be delivered to each location, and solve the problem with a fleet of vehicles with limited capacity, and you have the "Vehicle

Routing Problem" (VRP) which requires different algorithms again. If each location has goods to be collected and delivered to another location, then we have the "Pick-up and Delivery Problem" (PDP) which is different again. Add time-windows and we have the PDPTW problem class, etc.

Each of the above problems has its own set of benchmarks and each has a small army of researchers exploring the space of algorithms to generate a solution that is better than any previously found for one or more problem instances in the benchmark set for that problem class.

Every real-world problem is different. The problems faced by any two logistics companies are different. Similarly any two manufacturers face different production scheduling problems. Consequently it is unlikely that the same algorithm works for both problems.

The development of the best optimisation algorithm for a real-world problem requires, therefore, not a standard product and not a fixed algorithm. Moreover developing an algorithm from scratch is time-consuming and costly: that's what researchers do.

Opturion has an optimisation platform that enables algorithms to be configured and combined for the problem at hand. The consequence is that an efficient, flexible, scalable algorithm can be constructed within a short time, even for complex applications with multiple awkward side-constraints and ad-hoc optimisation requirements.