Report – Automatic Visum Model Calibration

Andrew Clarry – Co-op student, BA Group

Table of Contents

1. The Purpose of Optimization

The motivating problem for the optimization work done so far is the proper calibration of traffic models. The original model of this problem was the Visum mesoscopic model of traffic around the Humber Bay Shores neighbourhood in Toronto, focused on a development project in the former Mr. Christie's Factory lands. This model was originally developed by Aecom Canada Ltd, and generated traffic based on an origin-destination matrix obtained from an even larger-scale model. Traffic counts were collected at internal sites in the model, giving an idea of what kind of volumes the roads in the network should carry in both the AM and PM scenarios of the model.

The problem at hand was developing a Visum model, based on the origin-destination (O-D) matrix for the area, which also corresponded to the actual traffic observed within the network. Simply put, the traffic needed to move through the network the way in which it actually does. To correctly corral traffic into following the correct paths, the model could be tuned through the parameters of the volume-delay function (VDF) of target links in the network. These functions describe the relationship between traffic and travel times, and so tuning their parameters gives a way to influence the choices of drivers in the model.

The goal of the optimization process was to develop a procedure which would take these targets, and a collection of links, and tune those links VDF parameters such that the model is as closer to the targets as possible. This was ultimately achieved quite well, although the process did have some drawbacks.

2. Overview of optimization methodology

Any optimization algorithm fundamentally tries to compute a set of parameters which produce an optimal value of one or more *objective functions.* These objective functions, sometimes called *fitness functions* or by other names, are single-valued functions representing the outcome of some process. There are two key quantities in any transportation model which we might want to optimize: travel time, and traffic/passenger volume. In the case of our Visum models, we want the model's travel times and traffic volumes to be as close to observed real-world values as possible. While there exist optimization methods which can target multiple objective functions (these tend to be called multiobjective or Pareto methods), the best process is to use an objective function which is a weighted sum of our targets. In the

case of the Visum models which were tested, this target was a weighted sum of volume-based GEH, and percent differences between modelled and observed travel times.

The Visum meso-scale model of the transportation network is highly complex, which presents some problems for more traditional optimization methods such as gradient descent or root-finding based algorithms. These tend to require assumptions such as smoothness of the relationship between inputs and outputs, the existence of and ability to express a form for the derivatives of the outputs with respect to the inputs, or even that there is only one local optimum – none of which are the case for the Visum network.

In cases where there may be many different *local* optima, a special class of optimization algorithms are needed to get a good sense of the actual global optimal parameter values (those that minimize our weighted sum of GEH and travel time differences). These are generally called global optimization methods, and they tend to not provide guarantees that the solution provided is actually the best achievable one. Instead, they use heuristics to try to get an answer which is good – hopefully very close to the true global optimum – in far less time than a more exhaustive search would take.

Genetic algorithms are one of the oldest of these global optimization methods. They are a heuristic which tries to imitate biological evolution by simulating successive "generations" of solutions. The algorithm starts by randomly generating a set of solutions, and each generation of the algorithm produces a new set of individuals according to the evaluation of the objective function. In traditional schemes, these individuals are either the "elite" of the previous generation (the top quantile of the previous generation – usually the top 5-20% of individuals), "children" of two individuals in the previous generation which performed well (individuals are more likely to be chosen the better their objective function value), or "mutations" where an individual's proposed solution is changed slightly from before. These all require the parameters in the problem to be expressed as "genes", similar to biological genes. Today, most numerical optimization methods consider each "gene" to be a single parameter for the objective function. Thus, a "mutation" is adding some small randomly generated vector to the vector which is the individual's "genome", while a "child" is produced by randomly selecting parameters from each parent.

Genetic algorithms have been very successfully applied to many applied mathematics problems, in fields as diverse as antenna design, scheduling, and machine learning. Their success partially comes from the fact that they don't require many assumptions about the domain in which they are optimizing – they are able to handle discrete choices and real-valued parameters equally well. However, genetic algorithms are seen with skepticism by some experts because they are a very abstract heuristic. It can be unclear if a genetic algorithm is handling a problem well, and they provide no guarantees about the optimality of the final solution they produce – a bad genetic algorithm could give an answer very far away from the actual optimum. An easy way to improve genetic algorithms is to increase the size of their population and their running time. Larger populations essentially allow the algorithm to explore more of the "fitness landscape" at once, preventing the process from getting stuck in some local optimum. And longer running times essentially give the algorithm more chances to find a good solution. A good heuristic to use is to have a population at least 8x the size of the parameter space.

A similar method to genetic algorithms is Particle Swarm Optimization (PSO). This heuristic uses a population of individual solutions, similar to the genetic algorithm, but instead treats them as dynamic particles with momentum. These particles have forces applied to them in accordance with the fitness landscape that has been explored – they are attracted to the current global optimum found by the entire population, a more local optimum found within a local "neighbourhood" of points, and the best value that particle itself encounters. The relative power of these forces can be tuned as parameters for the PSO algorithm itself, which are referred to as *hyperparameters* in the optimization community.

Particle swarm optimization shares many strengths and weaknesses with genetic algorithms. As a heuristic approach, there are no guarantees that the procedure will find or even be close to the global optimum of the problem. As such, a procedure with poorly-tuned hyperparameters can perform very poorly. However, PSO has shown itself to be superior to genetic algorithms for certain types of problems – the team at the University of Toronto responsible for GTAModel have noted that they abandoned genetic algorithms for PSO as a tool to optimize parameters in their model. Also, in contrast to genetic algorithms, PSO can only be applied to continuous-valued optimization problems.

The final relevant optimization method is called simulated annealing (SA). Simulated annealing keeps track of single solution and selects new solutions in a semi-greedy way. It does this by modifying the current solution by a small amount (adding a small random number to each parameter, or choosing a new type at random for some randomly selected set of parameters in the discrete case). It then compares this new solution to the existing solution: if the new solution performs better, it automatically selects it to go to the next round. If the new solution performs worse, there is still some chance that the solver will choose to move to this new solution. This chance is determined by how much worse the new solution is than the current one, and by how far along the process is – expressed by a number within the model referred to as the "annealing schedule". Essentially, as the SA process progresses, it becomes less and less likely for it to choose a worse solution, eventually only choosing the better solutions.

As with all these global optimization heuristics, SA does not have guarantees about its general performance – the algorithm may not perform particularly well. But it performs incredibly well in some situations, and can perform well on very poorly behaved fitness functions (e.g. one with many local optima, where the "neighbourhood" of the optimum is very small). In addition, the algorithm gets to work at once, while PSO and genetic algorithms take at least a few iterations to start moving to better solutions.

3. The state prior to work

The original formulation of the Visum parameter optimization problem was to choose a VDF for each link from a set of about 50 possible VDFs. Thus, the optimization procedure did not work by optimizing the VDF parameters per se, but by optimizing choices of VDF "types". This presented a number of potential problems – the first is that this combinatorial expression of the problem may be harder to optimize, since it is harder for the algorithm to make small changes which allow it to get to progressively better. The second is that these discrete VDF types might not accurately describe the "true" VDF for a specific link.

Prior to my working on the Visum model optimization process at BA, a small package of code had been developed which was able to optimize Volume Delay Function (VDF) parameters for the Visum model of the Park Lawn and Lake Shore project. This applied a genetic algorithm to the problem of assigning the correct "type" of VDF to each road link in the Visum model. These VDF types were chosen beforehand, and amounted to a grid of "reasonable" values of parameters for the widely-used LOHSE VDF function. The genetic algorithm would use these options to try to find an option for each link, which together would minimize a weighted combination of GEH and percent differences in volumes and travel times between the Visum model and real-world observations.

4. Experimentation

The link type-based genetic algorithm optimization did quite well, producing a simulated network whose volumes were much closer to actual observed volumes than the raw model. There were two

options for improving the optimization procedure: the first was to tune the genetic algorithm's parameters; the second was to experiment with a different optimization procedure altogether.

Initial work with tuning the parameters of the algorithm itself proved fairly fruitful. In particular, increasing the population size from 25 to 200 both improved the answer and dramatically increased the rate at which better solutions were found. This value of 200 is actually the minimum recommended in Mathworks' own documentation, which recommends at least 200 individuals for a problem with more than 6 parameters (the Park Lawn and Lake Shore model had 34 parameters). Unfortunately, Matlab's genetic algorithm solver is quite limited when it comes to these kind of integer optimization problems, where the parameters must be integers. While there are quite a few tuneable parameters in the Global Optimization Toolbox's genetic algorithm implementation, only a few of these can be tuned in integer optimization problems.

The limitation with integer optimization in Matlab's Global Optimization Toolbox prompted a move to expressing the problem in a continuous format. This essentially amounted to reformulating the code which passed parameters into Visum – the target links would each be assigned a unique Link Type within Visum, and the optimization code would pass parameters directly into the VDF parameters for each link type.

Changing to a continuous parameter version of the problem allowed the full power of Matlab's Global Optimization Toolbox to be leveraged. The genetic algorithm option continued to be explored, along with two other optimization strategies: Simulated Annealing, and Particle Swarm Optimization. PSO operates with a similar logic to the genetic algorithm, making use of a large number of individual solutions to concentrate on good solutions. Simulated annealing operates somewhat differently, instead being a single point which wanders around in the solution space in a semi-random manner.

Initial experimentation with PSO quickly found it to be a superior alternative to the genetic algorithm: under the same number of iterations (runs of the Visum model), it almost always outperformed the genetic algorithm. Even as the genetic algorithm's performance was improved as the algorithm's hyperparameters were explored in the continuous formulation of the problem, the PSO solution continued to outperform it.

5. Results

Ultimately, Particle Swarm Optimization was found to be the most powerful of the algorithms explored in optimizing VDF parameters to target link attributes. A large swarm size of 100 or more particles was found to perform quite well, and running the algorithm with parameters which favoured individual exploration improved solutions even more. An individual neighbourhood fraction of 15% (vs the default 25%) and a self adjustment weight of 2.0 and social adjustment weight of 1.0 gave better solutions which converged faster than the defaults. A niche was identified for the Simulated Annealing method, in quickly obtaining a solution that is fairly good. While PSO ultimately reaches better solutions than SA, SA improves fairly quickly. If a new set of parameters is needed within one or two hours, 500-600 iterations of SA can be run to get a fairly good solution.

However, a number of problems arose in actually applying the calibrated Visum model. While link traffic volumes were incredibly close to observed values, the actual flow of traffic in the model deviated significantly in some places. This manifested itself in turning movements at intersections being significantly off from observed values. Because the Humber Bay area is divided by the Gardiner Expressway and Lakeshore rail corridor, the turning movements at some key intersections – such as Windermere and The Queensway and Windermere and Lake Shore – actually had a significant effect on network performance. This was fixed somewhat by tuning special turning VDFs for left turns in this

Eastern half of the model, but needing to rely on new optimized parameters meant that it took several days to fix this particular problem.

	Original Model		PSO Optimized Model	
	AM	PM	AM	PM
GEH < 5	17	20	34	33
5 < GEH < 10	12	15	5	6
GEH > 10	10	4	0	0
Sum of GEH scores	1390	822	139	192
	2212		330	

GEH Comparison between pre-optimization model, and model after final recommended PSO optimization.

6. Next Steps

There are many directions which might be taken with this optimization strategy.

Early on in my work, it was discussed that this methodology could be used for parameter optimization in Vissim models as well. I would be cautious about engaging in this with the current optimization process. It appears that the optimization procedure can do a fairly good job of improving the fitness function by overfitting data, causing huge deviations in some model performance metrics which were not incorporated into the fitness function. A good test may be to apply the discrete optimization method to the Vissim model – for instance, choosing a small number of plausible driver behaviours, then assigning a number of high-volume corridors to specific link behaviour types, and then having the algorithm match link behaviour types to driver behaviours. While this will constrain the optimization algorithm's performance, the constraint might help prevent overfitting which would make the model poorly calibrated for actually predicting network performance. This constraining might also be done by incorporating a large number of targets, such as the distributions of queue lengths at intersections, and travel times for multiple turning movements.

Another problem which is anticipated in moving the procedure to Vissim is the length of time taken to evaluate the model. The Visum model takes an order of seconds for the fitness function to be evaluated, while a Vissim model will be on the order of minutes, if not tens of minutes for a large network with many vehicles. With dozens of parameters to optimize and the very chaotic fitness function, it may prove very difficult to do a comprehensive optimization of Vissim model parameters in this heuristic way. Keeping the number of parameters small will be a key, and – given the runtime of a Vissim model – techniques such as surrogate-based optimization may be useful in speeding up the optimization process' run time.

Applying Particle Swarm Optimization to link VDFs has proven to be incredibly powerful, but there is a risk that the process fits the model to the data too well. Making small changes to an optimized network – e.g. adding a few seconds of delay to a target turn, or making a small matrix adjustment – tends to result in significantly worse performance. This is a worrisome indication that the model does not actually reflect the real-world transportation network particularly well, and that it may have poor predictive power when applied to future traffic demand. This potential issue is further highlighted by the significantly off turning volumes predicted in the optimized Visum model – while link volumes and travel times are fit very well, other aspects of the network may not reflect real-world conditions.

This potential overfitting may be mitigated by adding additional targets to the optimization procedure. While the procedure may not be able to directly modify the parameters of turning movements, it should try its hardest to fit the model to our observations of turning movements. Not including these as targets leaves the procedure free to vary turning volumes to whatever it wants. The new method of configuring targets for the optimization procedure which I developed would make it very easy to add functionality to target turning movement volumes, and make it possible to target more complex values, such as travel time along a longer corridor than a single segment. Future experimentation with the optimization procedure should figure out the proper weights to put on these targets to balance model performance against overfitting.

A feature which was not sufficiently explored in my work was the integration of volume-delay relations in turning movements. I had explored a model where turns were categorized according to whether they were signalized or not, with capacities being assigned to each type and a single turn VDF being applied for each turn direction (left, right, and through). This produced a model which was significantly worse than the model without turn VDFs, and it was not explored further. However, having a concise set of turning movement capacities and VDF parameters would likely improve model performance at very little cost.

7. Conclusions

The experience I have had working on the optimization process makes it clear that model performance can be improved by automated parameter tuning. In particular, optimizing link VDF parameters with Particle Swarm Optimization using a population of 140 particles, self-adjustment weight of 2.0, and social adjustment weight of 1.0 consistently produces models which are very well calibrated to target link volumes in both the AM and PM. However, work still needs to be done to ensure these models are robust and actually capture the dynamics of the road network – this should include experimentation with turn delays, including other known values in the optimization procedure, and constraining multiple links to share the same VDF. Once good methods are found for ensuring the robustness of the optimized models, it should be possible to start applying optimization procedures to Vissim microsimulation models as well.