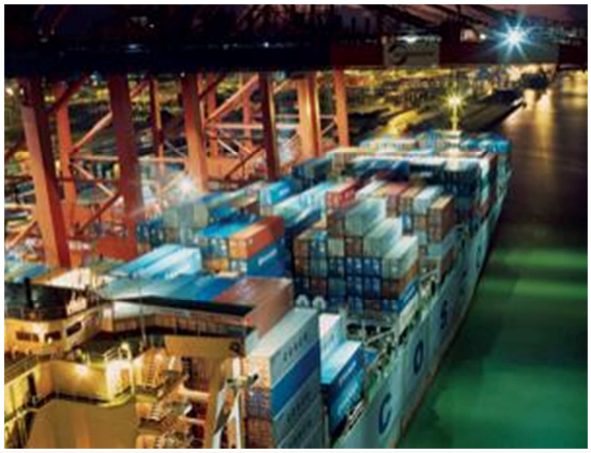
“The Algorithm That Runs the World”

August 20th, 2012 by Erwin Gianchandani

New Scientist published a great story last week describing the history and evolution of the simplex algorithm — complete with a table capturing “2000 years of algorithms”:

Its services are called upon thousands of times a second to ensure the world’s business runs smoothly — but are its mathematics as dependable as we thought?



YOU MIGHT not have heard of the algorithm that runs the world. Few people have, though it can determine much that goes on in our day-to-day lives: the food we have to eat, our schedule at work, when the train will come to take us there. Somewhere, in some server basement right now, it is probably working on some aspect of your life tomorrow, next week, in a year’s time.

Perhaps ignorance of the algorithm’s workings is bliss. The door to Plato’s Academy in ancient Athens is said to have borne the legend “let no one ignorant of geometry enter”. That was easy enough to say back then, when geometry was firmly grounded in the three dimensions of space our brains were built to cope with. But the algorithm operates in altogether higher planes. Four, five, thousands or even many millions of dimensions: these are the unimaginable spaces the algorithm’s series of mathematical instructions was devised to probe.

Perhaps, though, we should try a little harder to get our heads round it. Because powerful though it undoubtedly is, the algorithm is running into a spot of bother. Its mathematical underpinnings, though not yet structurally unsound, are beginning to crumble at the edges. With so much resting on it, the algorithm may not be quite as dependable as it once seemed [more following the link].

…

For today’s mathematicians, dimensions are not just about space. True, the concept arose because we have three coordinates of location that can vary independently: up-down, left-right and forwards-backwards. Throw in time, and you have a fourth “dimension” that works very similarly, apart from the inexplicable fact that we can move through it in only one direction.

But beyond motion, we often encounter real-world situations where we can vary many more than four things independently. Suppose, for instance, you are making a sandwich for lunch. Your fridge contains 10 ingredients that can be used in varying quantities: cheese, chutney, tuna, tomatoes, eggs, butter, mustard, mayonnaise, lettuce, hummus. These ingredients are nothing other than the dimensions of a sandwich-making problem. This can be treated

geometrically: combine your choice of ingredients in any particular way, and your completed snack is represented by a single point in a 10-dimensional space.

In this multidimensional space, we are unlikely to have unlimited freedom of movement. There might be only two mouldering hunks of cheese in the fridge, for instance, or the merest of scrapings at the bottom of the mayonnaise jar. Our personal preferences might supply other, more subtle constraints to our sandwich-making problem: an eye on the calories, perhaps, or a desire not to mix tuna and hummus. Each of these constraints represents a boundary to our multidimensional space beyond which we cannot move. Our resources and preferences in effect construct a multidimensional polytope through which we must navigate towards our perfect sandwich.

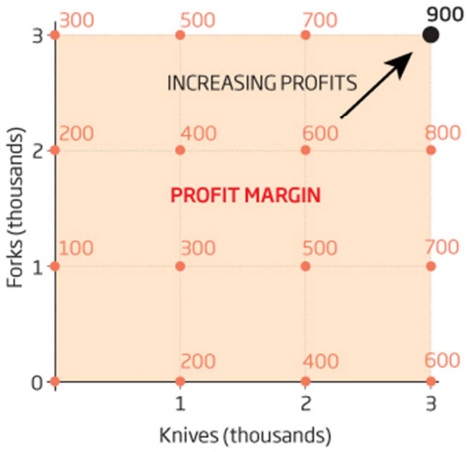
In reality, the decision-making processes in our sandwich-making are liable to be a little haphazard; with just a few variables to consider, and mere gastric satisfaction riding on the outcome, that's not such a problem. But in business, government and science, similar optimisation problems crop up everywhere and quickly morph into brutes with many thousands or even millions of variables and constraints. A fruit importer might have a 1000-dimensional problem to deal with, for instance, shipping bananas from five distribution centres storing varying numbers of fruit to 200 shops with different numbers in demand. How many items of fruit should be sent from which centres to which shops while minimising total transport costs?

A fund manager might similarly want to arrange a portfolio optimally to balance risk and expected return over a range of stocks; a railway timetabler to decide how best to roster staff and trains; or a factory or hospital manager to work out how to juggle finite machine resources or ward space. Each such problem can be depicted as a geometrical shape whose number of dimensions is the number of variables in the problem, and whose

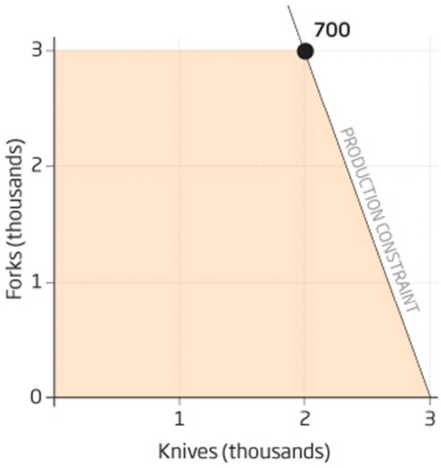
Room for improvement ©NewScientist

Many business problems can be reduced to patterns in geometry – as this simple 2D example shows

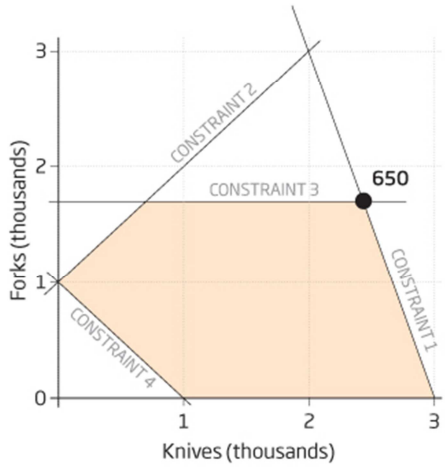
A cutlery factory makes $200 profit for every 1000 knives and $100 for every 1000 forks. With no constraints on production, **more profit** is made by making more of both



In the real world, finite staff or machine resources will mean the more forks you make, the fewer knives you can make. That constrains your operating space and the **maximum profit** you can make



Further constraints, such as demand for cutlery, restrict your operating space to a 2D geometric shape – and the **maximum achievable profit** always lies at a corner of that shape



boundaries are delineated by whatever constraints there are (see diagram). In each case, we need to box our way through this polytope towards its optimal point.

This is the job of the algorithm.

Its full name is the simplex algorithm, and it emerged in the late 1940s from the work of the US mathematician ***George Dantzig****, who had spent the second world war investigating ways to increase the logistical efficiency of the U.S. air force. Dantzig was a pioneer in the field of what he called linear programming, which uses the mathematics of multidimensional polytopes to solve optimisation problems. One of the first insights he arrived at was that the optimum value of the "target function" — the thing we want to maximise or minimise, be that profit, travelling time or whatever — is guaranteed to lie at one of the corners of the polytope. This instantly makes things much more tractable: there are infinitely many points within any polytope, but only ever a finite number of corners.*

If we have just a few dimensions and constraints to play with, this fact is all we need. We can feel our way along the edges of the polytope, testing the value of the target function at every corner until we find its sweet spot. But things rapidly escalate. Even just a 10-dimensional problem with 50 constraints — perhaps trying to assign a schedule of work to 10 people with different expertise and time constraints — may already land us with several billion corners to try out.

The simplex algorithm finds a quicker way through. Rather than randomly wandering along a polytope's edges, it implements a "pivot rule" at each corner. Subtly different variations of this pivot rule exist in different implementations of the algorithm, but often it involves picking the edge along which the target function descends most steeply, thus ensuring each step takes us nearer the optimal value. When a corner is found where no further descent is possible, we know we have arrived at the optimal point.

Practical experience shows that the simplex method is generally a very slick problem-solver indeed, typically reaching an optimum solution after a number of pivots comparable to the number of dimensions in the problem. That means a likely maximum of a few hundred steps to solve a 50-dimensional problem, rather than billions with a suck-it-and-see approach. Such a running time is said to be "polynomial" or simply "P", the benchmark for practical algorithms that have to run on finite processors in the real world.

Dantzig's algorithm saw its first commercial application in 1952, when Abraham Charnes and William Cooper at what is now Carnegie Mellon University in Pittsburgh, Pennsylvania, teamed up with Robert Mellon at the Gulf Oil Company to discover how best to blend available stocks of four different petroleum products into an aviation fuel with an optimal octane level.

Since then the simplex algorithm has steadily conquered the world, embedded both in commercial optimisation packages and bespoke software products. Wherever anyone is trying to solve a large-scale optimisation problem, the chances are that some computer chip is humming away to its tune. "Probably tens or hundreds of thousands of calls of the simplex

method are made every minute," says Jacek Gondzio, an optimisation specialist at the University of Edinburgh, UK…

And here's the table:

2000 years of algorithms

George Dantzig's simplex algorithm has a claim to be the world's most significant (see main story). But algorithms go back far further.

***c. 300 BC** THE EUCLIDEAN ALGORITHM*

From Euclid's mathematical primer Elements, this is the grandaddy of all algorithms, showing how, given two numbers, you can find the largest number that divides into both. It has still not been bettered.

***820** THE QUADRATIC ALGORITHM*

The word algorithm is derived from the name of the Persian mathematician Al-Khwarizmi. Experienced practitioners today perform his algorithm for solving quadratic equations (those containing an x^2 term) in their heads. For everyone else, modern algebra provides the formula familiar from school.

***1936** THE UNIVERSAL TURING MACHINE*

The British mathematician Alan Turing equated algorithms with mechanical processes — and found one to mimic all the others, the theoretical template for the programmable computer.

***1946** THE MONTE CARLO METHOD*

When your problem is just too hard to solve directly, enter the casino of chance. John von Neumann, Stanislaw Ulam and Nicholas Metropolis's Monte Carlo algorithm taught us how to play and win.

***1957** THE FORTRAN COMPILER*

Programming was a fiddly, laborious job until an IBM team led by John Backus invented the first high-level programming language, Fortran. At the centre is the compiler: the algorithm which converts the programmer's instructions into machine code.

***1962** QUICKSORT*

Extracting a word from the right place in a dictionary is an easy task; putting all the words in the right order in the first place is not. The British mathematician Tony Hoare provided the recipe, now an essential tool in managing databases of all kinds.

***1965** THE FAST FOURIER TRANSFORM*

Much digital technology depends on breaking down irregular signals into their pure sine-wave components — making James Cooley and John Tukey's algorithm one of the world's most widely used.

***1994** SHOR'S ALGORITHM*

Bell Labs's Peter Shor found a new, fast algorithm for splitting a whole number into its constituent primes – but it could only be performed by a quantum computer. If ever implemented on a large scale, it would nullify almost all modern internet security.

***1998** PAGERANK*

The internet's vast repository of information would be of little use without a way to search it. Stanford University's Sergey Brin and Larry Page found a way to assign a rank to every web page – and the founders of Google have been living off it ever since.

Check out the full story on the *New Scientist* website (accessible for the next four days to free subscribers).
(Contributed by Erwin Gianchandani, CCC Director)