computing@tanet.edu.te.ua
www.tanet.edu.te.ua/computing

ISSN 1727-6209
International Scientific
Journal of Computing

INDUSTRIAL APPLICATIONS OF COMPUTATIONAL INTELLIGENCE

Colin M. Frayn [1)]

[1)] CERCIA, School of Computer Science, University of Birmingham, Edgbaston, Birmingham, B15 2TT, UK.
cmf@cercia.ac.uk. http://www.cs.bham.ac.uk/~cmf/

Abstract: *In this paper, I cover in more detail two specific applications where Computational Intelligence systems have been used in industry. In particular, I consider the problems of path optimization through an inhomogeneous road network, and data analysis for financial applications. This paper will deal with several important questions about the applicability of those techniques in real-world scenarios, and will show how some of these issues have been directly addressed in order to create value for our business partners.*

Keywords: *Natural computation; industrial applications; optimization; data mining; computational intelligence ;evolutionary algorithms*

1. INTRODUCTION

Computational Intelligence techniques have existed for several decades, but are only now being seriously applied to real-world industrial problems. In previous work [1], I have given a general overview of a number of projects in progress in this area. In this paper, I will expand on some of those, giving a more detailed analysis of technologies in development.

Computational Intelligence techniques are usually conceptually straightforward, with very little or no complicated analysis required before applying them to an industrial problem. However, the skill involved in applying these techniques accurately lies in knowing which techniques are suitable for which problem, and how can one tune and optimise the algorithms used in order to avoid the issue of suboptimal or misleading performance.

So the skill in applying Natural Computation (NC) algorithms lies not in extended mathematical analysis, nor in exceptional programming expertise, but rather in the thorough understanding of one's problem space; in the intuitive comprehension of the inherent drawbacks and dangers of stochastic, population-based search; and in the perception necessary to recognise and diagnose potential systematic errors.

In this paper, I assume a basic understanding of NC techniques, including a familiarity with the concept of population-based stochastic search, and the components of a standard genetic algorithm. I will also assume some understanding of the problems to which we are applying these techniques, as these have been introduced in the previous paper in this series [1].

In section 2, I will cover once more the question of salting truck routing optimisation, which has direct applications in the field of route optimisation and constrained search. The third section will revisit the problem of data mining for financial analysis, which covers issues of data bias, incompleteness and efficient choice of representation. Section four summarises and concludes.

2. ROUTE OPTIMISATION FOR SALTING TRUCKS

In the first paper in this series [1], I introduced the issue of Capacitated Arc Routing within the optimization of routes through real-world transport networks. Our partners, Entice Technology Ltd., have developed an advanced prediction model to calculate road surface temperature based on climate models and geographical data, together with meteorological data gathered in real time from the government.

Our task was to design a system which took these data directly from their prediction model, and merged them with existing map data, available from the government transport authorities, in order to create the most efficient routes for salting trucks through this complex, constrained network.

Conventional route optimization techniques, for example A-star [2,3], rely on a depth-first search

through the available nodes which, as the network complexity increased, rapidly becomes prohibitive. In fact, the general family of these problems, known as the 'Travelling Salesman Problem' (TSP), belong to the class of NP-hard problems. That is to say, there is no known deterministic polynomial-time algorithm for solving these problems in a provably optimal manner. All solutions are non-deterministically polynomial at best in the number of nodes in the network under consideration and, perhaps more importantly, any problem in the class NP can be transformed in polynomial time into an instance of the TSP. This means that an efficient solution to the TSP would have important implications throughout computer science.

Analytical solutions, such as A-star and more specific solutions to the example of a closed loop path, such as 'cutting planes' [4] provide some powerful approaches to this problem. However, they are still NP in complexity, and are based on very fragile mathematics, which become completely invalidated by even the most trivial of constraints.

Real world problems very rarely (perhaps never) conform to clean, uncomplicated mathematics. For this reason, analytical solutions are rarely the best option for complex problems, especially those where constraints are likely to be difficult to translate into a tractable, analytical form.

Travelling Salesman problems may be formalized using a series of definitions. Firstly, we have a set E of edges, of cardinality N. These edges join together vertices (about which we are not concerned at this time). The edges have an associated cost, with the cost for edge 'n' denoted as C_n, in terms of distance and therefore time required to travel along them (which is expected to be proportional to the distance). We have a set T of trucks, with cardinality K, each of which has a maximum work capacity. The capacity for truck 'i' is denoted P_i. Often these capacities are the same, though here we consider the potential for variability.

In the conventional TSP formulation, we must visit a set of vertices once and once only each. In this particular salting truck application, we are concerned with visiting a set of edges, not the vertices, but the problem is directly equivalent.

Each truck then must visit an ordered list of edges, giving a total complexity (summing over trucks 'i' and edges 'n'):

$$Cost = \sum_{i=1}^{K} \sum_{n=1}^{N} \Delta_{ni} C_n \qquad (1)$$

Here, the function Δ_{ni} gives a value of 1 if truck 'i' visits edge 'n', and a zero otherwise. This sum is only valid if the following constraint holds:

$$\forall i, \sum_{n=1}^{N} \Delta_{in} C_n \leq P_i \qquad (2)$$

That is to say, no truck exceeds its maximum capacity. So far, we are only considering capacity as a general 'ability to work', supposing that a truck can travel a certain distance before having to stop, for whatever reason.

So far we have not considered any obligations on the edges which need to be salted, which would of course provide a further constraint.

Evolutionary framework

The TSP is tackled using an evolutionary algorithm with a number of important variations. Evolutionary algorithms are population-based techniques consisting of a population of encoded potential solutions, and three major functional components, namely selection, mutation and crossover.

An individual in the TSP is encoded as a vector of points, representing the order in which a certain set of destinations is to be visited. In this vector, technically a permutation of the available vertices, each value must appear exactly once and once only. This causes problems later when we consider the three evolutionary operators and how they may be applied to an order-based individual.

Selection is usually based on a 'fitness function', measuring the accuracy with which any individual solves a given problem. In this case, the fitness function is obvious, being a descending measure of the total cost of traversing the links given, with lower numbers being superior.

Mutation is the process by which candidate solutions are slightly altered in order to introduce new information in the population of individuals (which can be though of as our gene pool, or space of potential solution elements). For a route, this is not a difficult proposition, with several simple mutations conventionally used:

- Swap mutation: Swap the position of two randomly chosen elements in the genome
- Permutation mutation: Choose a short subsection of the genome, and permute the order of the elements within that subsection.
- Optimisation mutation: Choose a short subsection of the genome, and exhaustively search for the optimum (i.e. the shortest) permutation of the elements in this section.

Crossover is the process by which candidate 'parent' individuals are merged together to create

new individuals for the next generation. Conventionally, when dealing with individuals represented as vectors of values, it is straightforward simply to perform an N-point crossover, where the components of the resultant 'child' vectors are taken from each parent, alternating in sections between arbitrarily chosen loci.

However, for the TSP, this approach cannot be used as it may lead to 'illegal' solutions. For example, given the two (legal) tour vectors (1,2,3,4,5,6) and (6,5,4,3,2,1), a one-point crossover after the third digit would produce the two 'children' (1,2,3,3,2,1) and (6,5,4,4,5,6). These were produced by taking the first half of one vector and concatenating onto it the second half of the other vector. These offspring are both 'illegal' because they visit certain destinations more than once, and others not at all.

We can define various order-based crossover operators for the TSP, of which the simplest is perhaps to consider neighbours in each string and, starting with a random point, choose subsequent points at random (without replacement) from the pool of (yet unused) neighbours to that point from the parent genomes. In the case where there is no available subsequent point, a new (yet unused) point is chosen at random and the algorithm continues.

These steps outline a methodology by which the TSP may be solved using an evolutionary algorithm. However, the challenges are twofold, concerning not only the internal workings of the (analytically simple) theoretical TSP, but also dealing with the extra constraints that inevitably arise from a real-world application.

Real World Constraints

The challenge in this problem is in designing a route for a variable number of trucks to follow, in a timely and efficient manner. That efficiency is in terms of distance travelled, grit spread, and time taken to cover a sufficient fraction of the road network. There are a number of strong constraints on the road network, that we can denote more formally in the following (non-exhaustive) list:

Trucks only have a certain maximum fuel and grit load. So far we have only considered one single potential capacity for each truck, analogous to a fuel supply. However, in reality we must consider two separate costs for each edge, namely the amount of grit required for each edge, and the amount of fuel required in order to traverse it. Hence we can introduce a new capacity S_i as the maximum salt capacity that each truck is able to provide.

Certain roads will have more severe ice cover than others. This means that the cost of salting a road is not uniquely proportional to the length of that road. Or, to put this another way, the two costs (fuel vs. grit) are not directly proportional. In addition, some roads are steep, meaning that it is far easier for a truck to navigate them in one direction than in the other, suggesting that the cost matrix in this case should by asymmetric. That is to say $C_{n(AB)} \neq C_{n(BA)}$ in general, where A and B are the two vertices spanning this edge.

We may have to retrace our steps over already-salted routes. In this case, we obviously don't need to deposit salt a second time, so we need to consider which edges have already been salted (either by ourselves or by another truck).

One issue here is that of potential synchronicity. For example, the trucks will be operating in parallel, so we would need to consider, when truck A passes over edge X, whether or not truck B has already passed that way first, or whether truck B will arrive later. This depends on the sum of the costs to each of these trucks of the routes that they traverse before arriving at edge X.

In practice, we can ignore the effects of synchronicity here because we have no problem with two trucks occupying the same edge simultaneously, and the efficiency of the evolutionary algorithm we are to apply will penalize solutions for which two different trucks both salt the same edge.

Some roads only allow traffic to move in one direction. We mentioned the possibility of an asymmetric cost matrix for steep roads, but another situation in which this could cause problems in when a road only permits traffic in one direction, in which case we have two options:

Our first option is to alter our algorithm fundamentally so as to forbid the generation of solutions which incorporate travelling down a one-way road.

The second, more attractive, and far simpler solution is to alter the costs for one-way roads so that the cost for traversing it in the opposite direction is extremely high. This way, the core algorithm will remain unaltered, and will prune out any individuals which attempt to traverse a 'backwards' link automatically.

Some roads are more vital than others. Not only will some roads have more severe ice cover, and therefore necessitate more salt, and will require a more urgent treatment, but also some roads are intrinsically more important than others. For example, major inter-city highways are more

important than small residential streets.

Therefore there is an innate hierarchy in terms of the importance of each road. So we can define not only a cost function (as above), but also a benefit function, which measures how effectively we have salted the required roads. This seemingly straightforward constraint expands this problem to a multi-criterion optimization problem in which we have more than one goal which we need to trade-off against each other.

For example, for some governments, money may be tight, necessitating a more frugal approach to salting. One solution to this issue is to linearly combine the two objectives together with user-defined weights which quantify the degree of importance assigned to each objective. Another way to proceed is to investigate the family of multi-objective evolutionary algorithms such as SPEA, SPEA2 or similar [5,6] which allow a family of solutions to be evolved for a multi-objective problem, retaining an archive of optimally-diverse non-dominated solutions. That is, solutions which, when arranged in a partial ordering based on all fitness measures, are not absolutely inferior to any other solution. A formal treatment of this class of algorithms is beyond the scope of this paper.

Different roads have different speed limits. This means that the mapping from edge length to edge cost is further complicated. In addition, the speed limit for a road might depend on whether a salting truck has already covered that particular road, hinting that our fitness calculation is going to have to consider some level of chronology of the parallel salting operations.

Some roads are too narrow for the trucks to pass through. This means that certain potential routes are not feasible. The options here are to remove those edges from the network entirely, or simply to assign them extremely high costs. The latter is generally preferable, as it solves the problem with no complexities.

The trucks are not homogeneous. This is an important constraint, which implies several alterations need to be made to our algorithm. Firstly, trucks have varying capacities, which means that the value of P_i is not generally the same for any pair of trucks. This also goes for our second capacity value, related to the volume of salt that each truck carries, denoted as S_i (see above).

The fact that trucks are not the same means that the cost function of each edge will also depend on the truck traversing it. For example, some trucks will be too wide to traverse narrow country lanes, and may not have the necessary power to climb steep slopes. Some junctions may be impossible for larger trucks. So our cost function for each edge, C_n, now becomes two cost functions, based on the fuel and salt costs respectively, and each of these varies with truck number 'i'. So we have:

C_{ni} = Fuel cost for truck 'i' to traverse edge 'n'.
L_{ni} = Salt cost for truck 'i' to traverse edge 'n'.

Note that each edge is now effectively a pair of edges, as we have to specify different costs from the two different directions. Therefore, the (directed) edge joining nodes A and B will not be the same as the edge joining nodes B and A. This cost matrix, giving the cost of traversing the edge between any two points, is now asymmetric, and three-dimensional (in order to accommodate variation by truck number). It is also sparse (in general, most points are not connected).

The updated constraints, after modifying equation (2), are shown below:

$$\forall i, \sum_{n=1}^{N} (\Delta_{ni} + \delta_{ni}) C_{ni} \leq P_i \qquad (3)$$

$$\forall i, \sum_{n=1}^{N} \Delta_{ni} L_{ni} \leq S_i \qquad (4)$$

Where here we have also defined δ_{ni}, as the number of times truck 'i' must visit edge 'n' *without* depositing salt. This allows us to deal with potential routes where a truck must return to its depot along roads it has already salted, or which have been salted by other trucks. Allowing routes to overlap is a very useful strategy, as it enormously simplifies the network required. In fact, in most cases it is absolutely necessary, especially if many trucks all depart from and return to the same depot.

Proposed Fitness Components

A proposed route must optimise most of the objectives that we have already mentioned. This gives us a convenient way to define our fitness function, F, for this constrained version of the TSP as a modification of equation (1):

$$F = \sum_{i=1}^{K} \sum_{n=1}^{N} \Delta_{ni} (C_{ni} + L_{ni}) + \delta_{ni} C_{ni} \qquad (5)$$

However, at this point our mathematical

treatment begins to break down as we consider the more hazily-defined objectives that we must now introduce into this formulation.

Minimise the total journey length, and thus minimize fuel cost. This has already been included into the above formula (5) as the sum of the components of the summation involving the fuel cost of an individual edge for a specified truck, C_{ni}.

Minimise the total journey time, and this grit the roads as quickly as possible. We haven't specifically considered journey time yet, given that the cost C_{ni} only tells us the *fuel* cost of traversing edge 'n' with truck 'i'. This may not be directly proportional to the *time* it takes to do so. For example, on roads where the speed limit is very slow, the fuel efficiency will be lower. We could introduce a further matrix here specifying the time cost for traversing edge 'n' with truck 'i'. However, we must also realize that this will, of course, depend on whether that edge has already been salted. So this takes us back to the point raised earlier about synchronicity of traversing edges that must be visited by two or more different trucks.

Clearly, it is becoming very difficult to insert this term into our well-behaved analytical fitness function, though this is not at all a problem for evolutionary computation. We can add in any term we wish into a fitness function, even if it involves complex calculations such as that described above.

Maximise the speed with which major roads are covered. As we mentioned earlier, some roads are more important than others, so we might want to reward any solution that approaches those roads earlier in the route in order to get them cleared as soon as possible. This can be done by adding in a scaled reward function into our fitness function, based on the importance of roads (which would be manually assigned) and the delay before which they are serviced.

Maximise the coverage of roads with the most severe ice risk, whilst minimizing the coverage of roads with lower risk. We have been given a series of surface temperature predictions from our partner company, which must also be added into this fitness function at this point. This can be done by biasing the importance of roads based on how likely they are to be icy.

One can also introduce the concept of obligatory edges – that is to say, edges which *must* be salted at some point. This can be done either formally (by prohibiting the production of any potential solution which doesn't cover all required edges). Or, far more clearly and simply, by increasing the importance of those edges by a large amount and penalizing heavily individuals which do not satisfactorily cover every single one.

Further details on our own approach to this problem can be found in our earlier review paper [1] and in papers by colleagues and collaborators [7,8,9,10].

3. STOCK FILTERING USING EVOLUTIONARY TECHNIQUES

In collaboration with the Investor's Chronicle (IC) magazine, Financial Times group, UK.

The Challenge

The problem formulation was introduced in my earlier paper [1]. In this paper, I will cover more of the algorithms involved in the analysis of financial data.

The main problem in the domain of financial analysis is in obtaining a sufficient data set. That is to say, we require a data set that is both unbiased and complete.

The problem with bias is a difficult one. For example, when selecting companies that exist in the top indexes today, such as the FTSE-100 in London, one is already selecting for companies with a past record of success. One is also skewing the analysis towards companies which have a large market capitalisation, and therefore companies within certain sectors. The FTSE-100, for example, is heavily biased towards energy stocks and banking.

However, in most cases, what we are interested in is not the absolute gain in a company's net enterprise value (EV), but rather the *percentage* gain, and how this is reflected in the stock price. Therefore, the absolute value of a company is of little interest, with the caveat that it must be large enough for its bid-offer spread (the gap between buying and selling prices for the stock) not to swamp any plausible gains, and also for a sufficient value of stock to be available.

Because of this goal, and also because of the limitations of efficiently-analysed companies (see 'Efficient Market Hypothesis' below), most of the work presented in this paper concerns a complete listing of all companies listed on the London Stock Exchange (LSE), limited only by a minimum market capitalisation of £10m, and a maximum bid-offer spread of 10% of the median share price.

In terms of completeness, our main problem is a pre-processing one. Not only are data predictably incomplete, due to the regular intrusion of public holidays and substantial political events (e.g. terrorist attacks, war etc.) but they are also rendered

incomplete by financial forces specific to each company.

Firstly, companies merge and de-merge, which complicates the data enormously. One must learn how to deal with such instances either by slicing the data at the point of merger, or else by ignoring the companies involved entirely. Often the latter is the best course of action, as mergers predictably cause abnormal stock behaviour that is perhaps not characteristic of the trends that we are trying to extract.

Secondly, companies start trading at different times. Some temporarily de-list from the stock exchange, and some permanently de-list or go bankrupt. This means that, for many companies, the time range over which we have satisfactory data varies.

Thirdly, companies often perform stock splits or consolidations, where the number of shares and the price per share are altered in a consistent way. For example, if a company issues one million shares at one pound each, and then the company's market capitalisation reaches 50 million pounds, the shares will now be worth 50 pounds each. It would be wise for that company to split its shares so that an individual share is not so valuable. The company could perform a 100:1 share split, increasing the number of issued shares to 100 million, and reducing the price-per-share by a factor of 100 to £0.50 to compensate. This would show up on the share price register as a sudden drop in share price of 100 times, but the market capitalisation of the company would remain constant.

One way to deal with this problem is to consider the market capitalisation itself, instead of the individual share price. The problem here is that market capitalisation data are generally harder to find than individual share prices. This problem is usually straightforward to correct for directly in the data, by matching sudden large price changes with news items from archived regulatory news announcements.

The Efficient Markets Hypothesis

The Efficient Markets Hypothesis (EMH) states that stock market prices already factor in all the available information about any stock at any time. Because of this, it is claimed that it is impossible to make any justifiable prediction about the future price movements of any stock without possessing information which has not yet been disclosed to the market as a whole.

Because of this, the majority of analytical work within major banks does not focus on spotting mid- to long-term trading opportunities, but rather on analysis of risk profiles, and also in micro-trading with a sub-second temporal resolution. These are beyond the scope of this paper.

However, the EMH is on shaky theoretical ground. The larger a company, the more analysts will be watching it in general. Large investment banks are only primarily interested in companies in which they can gain a sufficient shareholding to make a measurable absolute increase to their turnover. A bank worth £50Bn is unlikely to consider it worthwhile analyzing a textiles manufacturer worth only £10m, of which only a few tens of thousands of pounds worth of shares will be freely available at any one time.

This tends to suggest that the EMH will apply more accurately to larger companies, whereas within the ranks of smaller companies should lie a larger number of insufficiently-well analyzed companies which may give far greater possible returns.

Indeed, work by several researchers including Tsang et al. [11] shows that arbitrage opportunities *do* exist within the LSE, showing that the markets are still not completely efficient, even within the arena of exclusively large companies.

Long-Term Systematic Variation

Of fundamental interest in a study of computational intelligence applications within financial data mining is the simple underlying assumption that markets behave in a reasonably consistent, stable manner, even when given the effects of changing social, economic and technological influences.

Data mining, in the form covered here, concerns learning patterns from large volumes of past data, and using that knowledge to inform decisions about the future. For this to be a valid approach, one assumes that patterns detected in the past are *relevant* to the future.

This question reduces to two fundamental issues, both of which are worthy of further discussion.

Firstly, **to what degree is trading behaviour affected by predictable underlying human psychology and solid mathematical techniques?**

This is an important question, as it not only covers the question of applying predictive models trained on historic data to present and future market states, but it also applies to transferring extensively trained models amongst different stock exchanges.

If the behaviour of a market is largely dependent on the transient social and psychological whims of its traders, then one might expect that models trained on a British financial dataset may not apply in the US or Germany, for example. However, if that behaviour were based on solid, underlying mathematics and psychology (such as innate fears

and desires, which might be reasonably assumed to remain constant between nations) then we would expect that models trained at a different time or location may be applicable globally.

This also gives us an interesting avenue for study as nowadays, with the world of Internet investment, it is becoming more straightforward to invest in diverse markets around the world. Does this imply that the behaviour of all markets will become more uniform? To what degree do the underlying economic mechanisms in each country (tax rates, regulations, employment law) affect the stock prices, and to what extent are they affected by human social and psychological differences?

I believe that this is an open question, and an extremely interesting one which merits much further study.

Secondly, **are data mining methods learning real knowledge that depends on human trading behaviour, or are they merely learning to predict patterns in arbitrary time series?**

This is an important question, as it affects our likelihood to trust models generated from a data mining process. Is it *relevant* that markets work in a different way in Germany and Japan to Britain and the US? Can we still predict what the share price is going to do in each case purely based on solid signal-processing techniques? It may be that the underlying political, economic and sociological factors in each of these different countries do indeed alter the way company prices respond to expectations, but that may be inconsequential when one comes to predict those same price movements.

An analogy: If we were to analyse two dancers with completely different styles such as ballet and rock, we would agree that their movements are totally different, driven by different desires and portraying a different underlying scheme and psychology. However, if they were both to execute a jump in their routine, we could still predict when they would land. This is because we are trying to predict something based on the law of gravity – a law external to the superficialities of their unique dancing styles.

To return to the world of financial analysis, we could agree that companies in Britain may behave differently from companies in Japan, and even that banks might behave differently from software companies, but certain underlying economic laws still apply, such as those of supply and demand.

Genetic Programming for Stock Filtering

As explained in the first paper in this series [1], the technique of Genetic Programming (GP) is ideal for classification learning tasks such as stock filtering or individual stock analysis. Genetic Programs (GPs) are tree-based functional models which can be evolved in a standard evolutionary algorithm just as we can manipulate a vector of integers, or a bit-string.

Stock filtering involves reducing a large number of stocks in a statistically reliable way down to a far smaller subset which is expected to be likely to perform above average.

Stock filters, or 'screens', have been in existence for many years, and are usually based on solid economic rationalisation rather than anything from quantitative data mining. Examples are those screens by Benjamin Graham [12] designed to short-list stocks deemed safe and reliable for the risk-averse investor.

Genetic programming offers a powerful way to automate generation of stock screens using a blind, supervised technique that has no prior knowledge of investment techniques, and works purely from the data. One would expect that such a process could help derive investment strategies that are both reliable and accurate, possibly finding relationships that would be too counterintuitive for a human ever to investigate.

A detailed background to genetic programming can be found elsewhere, especially in Koza's foundational book [13]. Here, I want to discuss some of the potential pitfalls with this technique when applied to such a complex data mining problem. These issues generally fall into two separate categories, namely those concerning the fitness function, and those concerning the space complexity and the issue of diversity to avoid suboptimal maxima in the fitness landscape.

Fitness function issues are always a problem in any canonical genetic algorithm (GA). It is important that one chooses a fitness function that is accurately aligned with the outcomes that one wishes to obtain. Evolutionary algorithms will always find a way to exploit any poorly-considered fitness function in order to fulfil it to the letter, but with a potential solution tat does not achieve what you actually desire.

For example, a fitness function that measures, and aims to minimise, the number of trading errors placed by an automatic trader would be trivially satisfied by a trader that makes no trades whatsoever. Similarly, a fitness function that aims to spot the highest-gain stocks with great certainty would simply invest in *all* stocks, blindly, therefore guaranteeing success.

It should be clear that both of these scenarios, though literally fulfilling the fitness function exactly, are functionally useless in any trading environment.

So how does one specify a fitness function that

will drive evolved solutions towards ends that are actually valuable to the user? One possibility is to create a fitness function as a simulation. That is to say, start with a pot of money on day one, and follow exactly what your evolved individual says to do for (say) one year, and tally up the total value of your portfolio at the end. This, as it turns out, is a rather useful way to proceed, though the main problem is that it learns very little when faced with a monotonically increasing or decreasing index. One must also remember to factor in trading costs and bid-offer spread into the simulation, in order to discourage rash buying and selling over short timescales.

One further implementation problem with this scenario is that the GP individual will usually not be able to give a degree of confidence in its prediction. So when a GP says 'buy', one must decide what this is to mean, in terms of the simulation. For example, one could implement a standard iterative scenario, where 'buy' means 'buy one unit' (provided you have enough spare cash) and 'sell' means 'sell one unit' (provided you own some).

The GP would be trained to give one numerical outcome which would imply 'buy' if the value is positive, and 'sell' if the value is negative. Alternatively, one could implement a three-signal scenario where GP values within a certain tolerance of zero are treated as 'neutral' or 'no-operation (NOP)' signals.

The drawback of these simple strategies is that they are unable to take advantage of epochs in which a rapid sale or purchase of a large number of shares is clearly indicated, such as directly before an imminent market crash.

An extension to this two-signal (buy/sell) or three-signal (buy/neutral/sell) strategy is a five-signal strategy in which some degree of severity is offered by the GP. In this case, in addition to the three signals explained above, signals of high absolute magnitude would imply that the maximum possible quantity should be sold or bought. This leads to a 5-signal (buy-all/buy-one/neutral/sell-one/sell-all) strategy. In an extension, one could design a scheme where the quantity to buy or sell is proportional to the output of the GP.

The main drawback of this method is simply that the GP is encouraged to act severely, very rarely taking the cautious route. For example, if a positive gain can be achieved by playing cautiously, then a far greater positive gain can be achieved by trading on the same occasions, but more recklessly. The problem here being that the GP may well take advantage of an excellent buying opportunity early on, and then find a superior one without an intervening selling opportunity, and be unable to take advantage of it. This can be partly moderated by the introduction of checks allowing only a maximum value of shares to be traded on any one day, but this strategy too is not without drawbacks.

Further strategies, each with their advantages and disadvantages, have been investigated. The conclusion is that it is very difficult to specify an accurate fitness function which will drive the evolution of an ideal GP solution. As in so many other data mining applications, the conscientious researcher is encouraged to test any proposed algorithm on a wide variety of unseen data before claiming wildly favourable results.

The fitness landscape in most real world problems is a great deal more complex than in simple, analytical test problems. Share-price data are by no means an exception to this rule. One of the major problems is that a lot of naïve methods work reasonably well, and this means that the fitness landscape is littered with a large number of suboptimal maxima.

Another problem is simply that the fraction of the available solution space that is valuable is actually extremely tiny: most possible filters are average or worse. A random filter is likely to (approximately) find companies that track the index. So differentiating between poor, average and good filters is not straightforward. The dynamic range is not very large (a few percentage points is a big improvement in finance) and the fraction we're looking for is tiny. As opposed to the previous example of route optimisation where the difference between a poor and a good route is very clear and substantial.

All these points mean that the power of evolutionary computation – monotonically urging the population of filters towards greater and greater fitness – is substantially reduced. Without a clear and sharp fitness gradient, evolutionary search becomes far more stochastic in nature.

Given all these caveats, it is surprising that a GP approach to stock filters works at all. However, moderately successful GP filters can be evolved in a relatively short time, and they tend to select companies which are not, at first glance at least, unreasonable. Overwhelmingly, these filters pick out companies with high recent relative strength (that is, substantial share price growth relative to the index) and a sensible PE ratio (indicating that the stock is relatively cheap compared to its earnings potential).

This technique, known as momentum investing, tends to be successful primarily in bull markets – where the share prices are, on the whole, moving upwards. However, it is a generally applicable algorithm, making sound investment sense in a variety of economic climates.

Evolutionary Conjunctive Rules Algorithm

One of the drawbacks of GP is that, because of its great flexibility and ability to model such a complex family of functions, it is unfortunately a very slow algorithm, attempting to cover an enormous, heavily-exponential search space. It is also very susceptible to suboptimal local maxima, becoming stuck in areas of solution space that are not close to optimal globally.

In order to deal with this issue, and thanks to the 'no free lunch' theorem[14,15], we can gain speed and reliability only at the cost of losing one of GPs virtues, namely its extraordinary flexibility. One such compromise is achieved by using Evolutionary Conjunctive Rules (ECR).

ECR involves evolving a set of rules which are joined by 'AND' statements. Companies matching all of these rules with no exceptions, are allowed to pass the filter. All others are rejected. Shorter and more economical rule sets are preferred.

The genome encoding such rule sets is a variable length hierarchical genome, consisting of one or more rule entities. Each rule consists of a rule type ('greater than', 'less than', 'equals' etc.), a variable on which this rule applies, and a comparison value. In some cases, two values are required. For example, for the rule 'X < variable < Y'.

In addition to a genotype-phenotype encoding, the three obligatory components of an evolutionary algorithm, as before, are selection, mutation and crossover.

Mutations are defined straightforwardly. A mutation can entire delete an existing rule entity (if there exists currently more than one), add a new one (generated randomly) or alter an existing one. Alterations are applied to any of the variables in each rule entity, either altering the rule type, the variable to which it is applied, or the comparison value(s).

Crossover is also reasonably straightforward. As the ordering of the rules does not matter, then one can simply take two parents and retain each of their component rules with 50% probability. Some degree of mutation can, and should, also be applied during the crossover. This tends to produce offspring with a length (number of rules) equal to the average of the two parents, with a maximum length equal to the sum of the parents' lengths. There is a possibility of rejecting all of the parents' rules, in which case one could randomly select one of the parents' component rules to retain.

Selection is based on the standard tournament selection technique, with a fitness derived from the accuracy of the rule set when analysed on historical data. This accuracy can consist of several components, and it makes sense to include at minimum the following:

- A weighted 'error' count, including false and true positives, together with false and true negatives, differently weighted. Generally in financial applications (in contrast with, for example, medical applications), a false positive is far worse than a false negative. These weights can be tuned to suit the preferences of an individual investor, essentially quantifying his or her desired risk profile.
- A measure of the efficiency of the rule set, penalising the fitness for each extra rule required, and for unnecessarily large bounds on each rule entity. This is necessary to keep the search space tight, and to reduce unnecessary genome bloat.
- A scaled fitness, considering errors more influential if they apply to companies whose stock movements were more drastic during the training period.

Addendum: Work in Progress

There are many potential extensions to this work. Most interestingly, the potential to analyze how predictive models for company success vary across sector type. For example, would a model trained for financial service providers also work on automotive companies? This reduces once more to the same question of the universality of predictive models across varying scenarios.

As another interesting extension, it may be possible to introduce extra information from different data streams into the predictive model. For example, textual news sources and company results. Also, the performance of other companies in the same sector or market.

The potential for analyzing textual information is exciting. The main drawback of the stock filtering methods described above is that they cannot possibly take into account that which they do not know, such as news stories or global economic trends. We aimed to add into our models measures of external economic factors such as interest rates and inflation, though it would be possible to derive far more measures if we were to analyze news stories in more depth. For example, oil stocks seem to move in correlation with the degree of political instability in the world, especially in the Middle East.

In addition to global economic factors, analysis of text would also allow us to incorporate into our analysis a number of factors specific to the companies under scrutiny, for example share

purchases or sales made by company directors.

4. CONCLUSION

In this paper I have covered in more detail a selection of algorithms from the field of Computational Intelligence, which are being used with great success in industrial applications.

Specifically, I have covered algorithms for the optimisation of complex routing problems, with real world constraints. I have also discussed algorithms for analysis of financial data, mentioning two by name: Genetic Programming (GP) and Evolutionary Conjunctive Rules (ECR) techniques.

I have covered, in each case, a discussion of the pitfalls inherent in these techniques, together with suggested methods for dealing with these potential problems.

The field of Computational Intelligence is becoming increasingly popular within industry as a large number of difficult, computationally intensive problems present themselves and human intuition is no longer sufficient to comprehend or resolve such complex situations manually.

Computational Intelligence techniques give us the ability to solve these problems in reasonable time, with a minimum of human specialist expertise or technical analysis. They are also extremely flexible, allowing us to apply them to real world problems including difficult constraints, with only minor modifications.

5. ACKNOWLEDGMENTS

CMF acknowledges financial support and assistance from Advantage West Midlands, UK. Also from the companies and colleagues named in this report. Some of these projects were carried out with financial support from VIN Technology services, University of Birmingham, UK.

6. REFERENCES

[1] C. Frayn, “A Review of Industrial Applications of Computational Intelligence”, 2006, ISJC, In Press.

[2] P.E. Hart, N.J. Nilsson, B. Raphael, “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”. 1968, IEEE Transactions on Systems Science and Cybernetics SSC4 (2): pp. 100–107.

[3] P.E. Hart, N.J. Nilsson, B. Raphael, “Correction to ‘A Formal Basis for the Heuristic Determination of Minimum Cost Paths’”, 1972 SIGART Newsletter 37: pp. 28–29

[4] G.B. Dantzig, R. Fulkerson, S.M. Johnson, “Solution of a large-scale traveling salesman problem”, Operations Research 2 1954, pp. 393-410.

[5] E. Zitzler. Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. Ph.D. thesis, Shaker Verlag, Aachen, Germany, 1999.

[6] E. Zitzler, M. Laumanns, L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm", 2002, Tech. Report 103, Computer Engineering and Networks Laboratory, ETH Zurich.

[7] H. Handa, L. Chapman, X. Yao, "Robust route optimisation for gritting/salting trucks: A CERCIA experience", 2006 IEEE Computational Intelligence Magazine, Vol. 1, No. 1, pp6-9.

[8] H. Handa, L. Chapman, X. Yao, "Dynamic Salting Route Optimisation using Evolutionary Computation", Proc. CEC, 2005.

[9] L. Chapman, J.E. Thornes, A.V. Bradley, 2001, "Modelling of road surface temperature from a geographical parameter database. Part 1: Statistical," Meteorological Applications, Vol. 8, pp. 409–419

[10] L. Chapman, J.E. Thornes, A.V. Bradley, 2001, "Modelling of road surface temperature from a geographical parameter database. Part 2: Numerical," Meteorological Applications, Vol. 8, pp. 421–436

[11] E.P.K. Tsang, S. Markose, H. Er, "Chance Discovery in Stock Index Option and Futures Arbitrage," New Mathematics and Natural Computation, Vol 1, No 3, November, 2005.

[12] B. Graham, "The Intelligent Investor, Revised", 1972, Jason Zweig, Collins.

[13] J. Koza, "Genetic Programming: On the Programming of Computers by Means of Natural Selection", 1992, MIT Press.

[14] D.H. Wolpert, W.G. Macready, 1995, No Free Lunch Theorems for Search, Technical Report SFI-TR-95-02-010 (Santa Fe Institute).

[15] D.H. Wolpert, W.G. Macready, 1997, No Free Lunch Theorems for Optimization, IEEE Transactions on Evolutionary Computation 1, 67.

Colin Michael Frayn *obtained his undergraduate and PhD. degrees in astrophysics from the Institute of Astronomy, Cambridge University, UK. He now works at the Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), based at the School of Computer Science, University of Birmingham, UK. He is currently a senior research fellow, with interests in all aspects of Natural Computation. His most recent research is focused on genetic programming, financial modelling, data mining, visualization, virtual economies and game AI.*