Nonlinearity Linkage Detection for Financial Time Series Analysis

Theodore Chiotis
University College London
Gower Street, London, WC1E 6BT

Christopher D. Clack
University College London
Gower Street, London, WC1E 6BT
c.clack@cs.ucl.ac.uk

ABSTRACT

Standard detection algorithms for nonlinearity linkage fail when applied to typical problems in the analysis of financial time-series data. We explain how this failure arises when standard algorithms are applied naïvely, how linkage detection needs to be applied directly to the observed data samples, and how this raises problems that are not addressed by current techniques.

We extend the existing DSMDGA linkage detection algorithm and present a new system that can determine the required nonlinearity linkage in observed data samples for financial time series. The new system has been evaluated on synthetic datasets and experimental results are provided. The sensitivity of the system to changes in both the problem and the algorithm parameters has also been explored and we discuss the results. We present evidence of the success of the new system and identify areas for further work.

Categories and Subject Descriptors

I.2.M [**Artificial Intelligence**]: Miscellaneous

General Terms

Algorithms, Experimentation

Keywords

Genetic Algorithms, Linkage Learning, Epistasis, Composition, Perturbation, Hierarchical, Time Series, Finance

1. INTRODUCTION

A typical Genetic Algorithm (GA) or Genetic Programming (GP) problem in financial time-series analysis is to discover an unknown, often non-linear, relationship between a number of "predictor" time series and a "target" time series. These series are all real-valued and are collections of discrete observations. An example of such a problem would be the search for relationships between time series of economic factors (interest rates, gross domestic product, energy prices) and the performance of a stock index such as the FTSE100. It is assumed that the extent of the time series is limited to a period during which non-linear relationships are unlikely to vary. The search space is immense and the performance of evolutionary search would be substantially improved if non-linear and linear relationships between the predictor series with respect to the target series could be identified, since this would permit independent evolutionary search to be conducted on non-linear blocks of the chromosome, which may later be combined linearly.

Although our work is driven by the need to analyse financial time series data, the proposed new system is relevant to the analysis of more general GA or GP systems where fitness is an error measure derived from training data that comprises discrete, real-valued observations.

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

1.1 The problem

Consider Figure 1: we wish to find the unknown function that takes as input the values of the four predictor series and produces as output the corresponding value of the target series. We will use evolutionary search to do this, and the fitness function will simply apply the function suggested by an individual to each set of predictor values and return the square root of the average squared error from the desired target values.

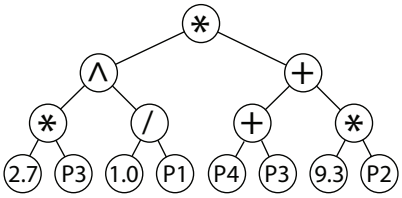
Time:	t_0	t_1	t_2	t_3	t_4	t_5
Predictor 1:	0.10	0.20	3.50	2.30	2.20	0.01
Predictor 2:	0.01	0.10	2.30	2.20	3.50	3.50
Predictor 3:	3.50	2.10	3.50	2.20	2.30	0.01
Predictor 4:	2.10	0.01	0.01	3.50	2.30	2.10
Target:	7.45	2.02	805.04	513.70	775.29	3.52

Figure 1: Example financial time series data comprising six observations of four predictors and one target series.

For this example, we assume that the desired function can be expressed as full tree of depth four; a GA representation might therefore be a vector of length 15 where the first seven loci are the dyadic arithmetic operators in the internal nodes and the final eight loci are the leaves that consist of either predictor identifiers or real-valued coefficients. See Figure 2.

This problem is very different from, for example, finding the optimal coefficients to a known equation. Here the equation is not known in advance, the representation is complex, and the search space is very large.

Information about linkage could potentially help to reduce the search space. If we could determine, for example, that predictors 1 and 4 were a linkage group, linearly separable from a second linkage group comprising predictors 2 and 3,



*	^	+	*	/	+	*	2.7	P3	1.0	P1	P4	P3	9.3	P2

Figure 2: Example tree (above) and corresponding GA chromosome (below) for time series analysis. The alleles for internal nodes are drawn from an alphabet of standard dyadic arithmetic operators and the leaves are either reals or predictor identifiers.

then the operator at the top of the tree could be restricted to a linear operator such as +, and two separate searches could be undertaken for subtrees of depth 3 — the predictors 1 and 4 appearing in only the left subtree, and the predictors 2 and 3 appearing in only the right subtree. Thus, by determining information about the (non-)linearity of contributions from the predictor values to the target values, the search space would be substantially reduced.

Linkage learning by perturbation

Phenotypic linkage is a measure of properties of the interactions of genes' contributions to fitness. This contrasts with "inheritability linkage", which is interdependence of the genes' probability of being inherited. Phenotypic linkage is a property of the fitness landscape, whereas inheritability linkage is a property of the representation and genetic algorithm. Our research focuses on a specific type of phenotypic linkage — non-linearity linkage — where the contribution to fitness of two or more genes is not simply a linear sum of the individual contributions to fitness of each gene.

Perturbation linkage learners such as LINC [5, 1] and LINC-R [6] determine non-linearity linkage between two loci i and j in a chromosome by calculating the fitness differences for four points in the search space: a starting point given by an arbitrary individual I_{xy} where the two loci respectively have values x and y, and three further points derived by keeping the values of all loci the same except that either i is perturbed to have value $x + \delta x$, or j is perturbed to have value $y + \delta y$, or both are perturbed. In this paper we call these four points a "quadlet".

Let the fitness of an individual I be given by $f(I)$. If the fitness difference $(f(I_{(x+\delta x)(y+\delta y)}) - f(I_{xy}))$ is not equal to the sum of fitness differences $(f(I_{(x+\delta x)y}) - f(I_{xy}))$ and $(f(I_{x(y+\delta y)}) - f(I_{xy}))$ then the two loci are non-linearly linked. The amount by which the two sides in the above equation differ is called the linkage measure of the corresponding pair of loci.

LINC deals with binary alleles and uses a single quadlet to test for linkage with binary perturbations, i.e. xy values of 00, 01, 10 and 11. LINC-R deals with real-valued alleles and uses several (randomly selected) quadlets to sample the range of possible values at the two loci between observed upper and lower bounds of x and y. See Figure 3.

The wrong way

Continuing the example from Figure 1, to determine non-linearity linkage between Predictor 1 (P1) and Predictor 2 (P2) we might naïvely decide to create a quadlet as follows: the starting chromosome can be chosen arbitrarily, so we start from a chromosome that contains one instance each of P1, P2, P3 and P4; then to create the next three chromosomes for the quadlet we perturb the two identifiers P1 and P2 to become the value 0.0 — first just P1, then just P2, then both. An example quadlet is given below:

Individual I_1**:** $+, +, +, +, +, +, +, 1.0, \underline{\mathbf{P1}}, 1.0, \underline{\mathbf{P2}}, 1.0, P3, 1.0, P4$
— fitness 492.53

Individual I_2**:** $+, +, +, +, +, +, +, 1.0, \underline{\mathbf{0.0}}, 1.0, \underline{\mathbf{P2}}, 1.0, P3, 1.0, P4$
— fitness 494.42

Individual I_3**:** $+, +, +, +, +, +, +, 1.0, \underline{\mathbf{P1}}, 1.0, \underline{\mathbf{0.0}}, 1.0, P3, 1.0, P4$
— fitness 494.41

Individual I_4**:** $+, +, +, +, +, +, +, 1.0, \underline{\mathbf{0.0}}, 1.0, \underline{\mathbf{0.0}}, 1.0, P3, 1.0, P4$
— fitness 496.30

The linkage equation calculates $(f(I_1) - f(I_4)) - ((f(I_2) - f(I_4)) + (f(I_3) - f(I_4)))$, which gives the value 0.0 and indicates that predictors 1 and 2 are *not* non-linearly linked.[1] But that's wrong! The target function for our example is $100 * (P1 * P2) + (P3 * P4)$ and so we know that predictors 1 and 2 *are* non-linearly linked.

The problem with this approach is that it measures the wrong linkage information. In order to reduce the search space *by constraining the structure of the chromosome*, we need to derive information about the linearity of the contribution of the predictors to the target equation — however, the above approach measures the linearity of the contribution of the predictors to the fitness landscape of the proposed solutions (i.e. the (root mean square) errors of the individual chromosomes from the target equation).

Is it therefore true that for this particular kind of problem, where we are trying to evolve an approximation to an unknown target function by minimising errors, it is impossible to determine linkage information that will help us constrain the structure of the chromosome? Of course not! But we must work on the observed data samples directly.

Linkage learning from observed samples

If we want to determine the linkage of the predictors, in a situation where the function relating the predictors to the target is itself being evolved by minimisation of errors, then we must look at the observed time-series values directly. In this case, to determine linkage we need to find quadlets *of predictor observations* — for each observation, the "fitness" is the value of the corresponding target time series.

In the example given above there are far too few observations to find a suitable quadlet. Fortunately, in time-series analysis the series are typically quite long with hundreds or thousands of observations.

Two problems arise when performing linkage learning on observed values:

1. the fitness function is only partially defined — fitness values (from the target series) are only available for the

[1]There is a possibility of apparent non-linkage even where predictors are linked, so normally we would sample several quadlets for different schemata. However, for this example we will assume such an effect has not occurred.

observed set, and for any other collection of predictor values the fitness is undefined;

2. the predictor values are typically reals.

Since the predictors are reals, we might try to use LINC-R for the linkage learning (the upper and lower bounds for LINC-R's random perturbations can be derived from the observed values). LINC-R performs the quadlet test for several different perturbations δx and δy, in order to sample the search space between the four outer points — if any one perturbation shows non-linearity then the two loci are said to be non-linearly linked. See Figure 3.

Perturbing a data value x by an amount δx is only useful if the new value $x + \delta x$ exists in the observed data set (with all other loci kept the same). We must search the entire set of observed predictor values in order to find a match to each of the four required points in the required quadlet. However, where the predictor values are reals, even a few thousand observations may not be sufficient to find the desired quadlets — there may not be any observed points containing the required values!

Thus LINC-R fails because it is a *perturbation* technique that is unsuited to a fixed set of observations that cannot be perturbed.

An alternative approach is to use a Dependency Structure Matrix (DSM) as found in the DSM-Directed Genetic Algorithm (DSMDGA) [11, 10, 9] — this is attractive because it samples individuals that exist in the population rather than creating new individuals. It would appear to be appropriate for discovering linkage in observed time series data *except* for the fact that the DSMDGA assumes a binary domain, whereas time-series data is real-valued. The DSMDGA approach is explained in greater detail in Section 1.3.

Thus the DSMDGA as it stands cannot be applied to our problem; but we will investigate how it can be extended to address our problem.

1.2 Related Work

In the field of linkage learning, various techniques have been proposed for linkage identification. Below we present some techniques for linkage learning, the use of which, at least without important modifications, is precluded by the intricacies of the problem being studied.

Standard statistical measures of correlation. Standard correlation functions generally measure linear rather than nonlinear relationships between variables, though the coefficient of nonlinear correlation may give some information. However, these measures are used to determine correlation between two time series, rather than detect the interaction of two series with respect to a third. Therefore, while it would be possible to obtain correlation measures for our predictors based on the available time series data, this would not give us any information about linkage with respect to the target. In the wider scope of time series analysis, a result showing strong correlation between two predictors might lead to one of the two being dismissed as redundant, however this is beyond the scope of this paper.

Piecewise Interval Correlation by Iteration (PICI) [7, 8]. PICI calculates correlation coefficients among real-valued loci of individuals in the population, and identifies sets of loci with high correlation as linkage groups. However, as mentioned above, these mathematical correlations indicate linear relations rather than non-linear relations.

Limited probing [2]. The limited probing technique proposed by Heckendorn and Wright identifies linkage by calculating the Walsh coefficients of the fitness function. This requires multiple evaluations of the function for specific inputs, which is precluded by the fact that only a limited number of input-output pairs is available in a time series. Some of the required input-output pairs may appear in a given data set; however, due to the number of necessary evaluations, the probability of all necessary pairs appearing in a time series of limited length is low.

Random walk [3]. This method proposed by Li and Li detects linkage in binary chromosomes by a random walk method. It might be possible to extend it to real-valued chromosomes, however its potential usefulness with respect to the problem being studied is limited by an additional factor. Since the process described requires a number of random walks to be made, and only one group of time series is available to us, we would have to segment this dataset into a number of smaller datasets in order to represent additional random walks. This problem is worsened by the fact that the efficiency of the algorithm in its current form is highly sensitive to limited data availability. Finally, certain types of time series (such as financial time series) are rarely truly random and are most often the result of a Markov process, therefore the assumption that a sample from this problem can be used to represent a random walk would not hold.

The Dependency Structure Matrix Driven Genetic Algorithm (DSMDGA) [11, 10, 9]. The DSMDGA is an algorithm that detects linkage in binary chromosomes by utilising a Dependency Structure Matrix, which is a matrix that captures the linkage of all pairs of genes in a chromosome. The linkage is calculated using the nonlinearity check found in LINC, which requires a small amount of evaluations to test the relationship between two genes. The DSM is then translated into linkage groups using a Minimum Description Length metric. Because of the small number of fitness function evaluations necessary to detect linkage, we find this technique particularly interesting — the standard formulation fails for our problem, as explained in the previous section, but it is capable of extension. We will provide a more detailed description of the DSMDGA in Section 1.3.

1.3 Non-linearity linkage detection for financial time-series analysis

We wish to design a linkage-learner that operates directly on the observed data values of financial time series. The following issues arise:

- the predictor series may have different ranges;
- the fitness function is partially defined — the observed data are the only points at which fitness is known, and linear interpolation is often unsafe due to uneven sampling and excessive volatility; and
- the predictor series comprise real-valued data, which makes it extremely difficult to find suitable quadlets for linkage identification.

Our new system is based on, and extends, Yu et al's DSMDGA [11, 10, 9], and we start by describing the DSMDGA in more detail.

DSMDGA

The DSMDGA uses a symmetric Dependency Structure Matrix (DSM) whose dimension is determined by the number of genes in the chromosome of the problem being studied. If two genes p and q are linked, then the DSM will have the value 1 at positions (p, q) and (q, p). If they are not linked, those positions will both contain the value 0.

The DSM is based solely on chromosomes existing in the population; this is done to minimise the number of evaluations of the fitness in the overall genetic algorithm. The linkage identification is the same as LINC, however the test is performed on observations of schemata instances rather than perturbing individual members of the population. Thus, the algorithm does not evaluate arbitrarily defined chromosomes, rather relying only on those that are part of the current and previous populations.

To obtain a good description of the linkage situation under these conditions, the algorithm averages the fitness values of all observed schemata at each generation. These values are averaged again over generations in order to refine the calculated average fitness of each schema. A schema is a chromosome which keeps one or more of its genes fixed, while the rest are wildcards, i.e. can take any possible value.

After these average fitness values have been calculated, the standard LINC nonlinearity check is performed on suitable schemata (not individuals) to detect linkage between pairs of genes. LINC is used because it is assumed that the loci in the schemata that are fixed have values drawn from a binary alphabet. The fitness values however may be integers or reals. By calculating the absolute value of the difference between the two parts of the check's equation, a linkage measure is calculated and placed in the DSM.

The DSMDGA requires a binary classification of linkage (two genes are either linked or not linked) in order to drive the partitioning of the search space. Therefore, after the linkage measures for all possible pairs of genes have been calculated, a machine learning algorithm is required to separate these linkage measures into those that indicate nonlinearity linkage and those that don't. If a given pair of genes has a linkage measure that indicates linkage exists, the value 1 is placed at the appropriate positions in the DSM. Otherwise, the value 0 is used.

Problems with the DSMDGA

The DSMDGA faces the following problems when applied to observed financial time series data, all caused by the fact that the predictor values are reals:

- depending on the range and accuracy of these reals, there may be very few instances of exact repetition at one locus. This undermines the concept of taking an average fitness across samples of schemata;
- for the same reason, it will be extremely difficult to find suitable schema quadlets in the observed data;
- it will not be possible to apply the LINC non-linearity check (which assumes a binary alphabet), though LINC-R could be used instead.

1.3.1 The new linkage-learning system

Our system makes the following two major extensions to the DSMDGA:[2]

1. We extend the algorithm to discover linkage where the data values are drawn from a small, finite (non-binary) alphabet. The extension is designed to address the three problems outlined in the previous section, and is explained in more detail in the following section.

2. In order to make the predictor data values more amenable to linkage analysis, we apply a **quantisation** algorithm. This transforms the data values from reals into values drawn from a small, finite alphabet, suitable for analysis by our extended DSMDGA. If the alphabet is small, the linkage analysis will be easy but it will have poor accuracy: a larger alphabet increases accuracy, but makes analysis more difficult.

Linkage detection for finite non-binary alphabets

The DSMDGA assumes values are binary, and can apply the LINC test to the schemata quadlet $\{S_{00}, S_{01}, S_{10}, S_{11}\}$; as described above, the fitness values used in the calculation are averages across observed instances of each schema.

By contrast, where the values are drawn from a small finite alphabet — for example, $[a, b, c]$ corresponding to three quantisation bins — then to determine linkage between predictors P1 and P2 we must consider at least the two quadlets $\{S_{aa}, S_{ab}, S_{ba}, S_{bb}\}$ and $\{S_{aa}, S_{ac}, S_{ca}, S_{cc}\}$ and preferably also the three other quadlets in the plane: $\{S_{ba}, S_{ca}, S_{bb}, S_{cb}\}$, $\{S_{bb}, S_{cb}, S_{bc}, S_{cc}\}$ and $\{S_{ab}, S_{bb}, S_{ac}, S_{bc}\}$. The fitness values, as with the standard DSMDGA, are averages from all relevant schemata in the observed data. See Figure 3 (middle) for an example with an alphabet of four values $[a, b, c, d]$.

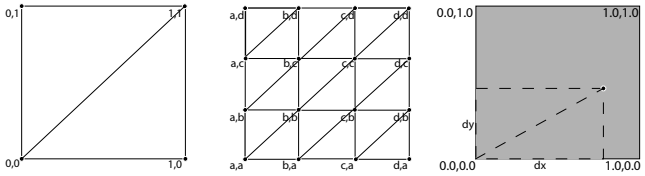


Figure 3: Linkage detection in three situations: with a binary alphabet (left), a small finite non-binary alphabet (middle), and a (normalised) continuous alphabet of reals (right). The first uses a single quadlet, the second either samples or uses all quadlets shown, and the third must sample from the potentially infinite number of quadlets.

Algorithm

The algorithm proceeds as follows:

1. All predictor data is normalised and quantised. This is done using a simple normalisation and rounding procedure: the values for each predictor are normalised independently of the other predictors to the interval [0, 1] and then that space is evenly separated into an arbitrary number of partitions. [3]

[2]The output of our analysis is always the binary DSM, which we inspect manually for linkage groups, so we do not implement an automatic grouping method.

[3]This is the simplest form of quantisation conceivable, but one that could separate values that might be best left in the same quantisation group. An example of this would be observing a time series such as (0.80, 0.89, 0.90). If we used the aforementioned method with 10 quantisation groups, 0.80 and 0.89 would

2. For all pairs of predictors (p,q):

   (a) for all possible quantum values i from predictor p and j from predictor q, viewed as schemata:

   - create a mean fitness matrix MF containing at each entry $MF(i,j)$ the mean fitness of all observed incidences of schema(i,j)

   Each entry $MF(i,j)$ corresponds to a specific schema which only holds stable the predictors being tested, and all the observed target values for each schema are averaged. The average value is then used to represent all instances of that particular schema in the algorithm.

   A square of four entries in this matrix corresponds to a "quadlet" suitable for the nonlinearity check. The use of this data structure greatly facilitates an exhaustive search for such groups of data points.

   (b) Initialise an empty linked-list L

   (c) For each entry (i,j) in $MF(i,j)$:

   - if (i,j) is the top left locus of a valid quadlet (i.e. four loci (i,j),(i+x,j)(i,j+y),(i+x,j+y) with non-zero values) then measure its linkage value and add that value to the linked-list L

   (d) Set the real-valued DSM matrix entry $DSMr(p,q)$ to be the average of the values held in the linked-list L

3. Analyze the $DSMr$ to calculate an appropriate threshold to be used to distinguish "linked" from "not linked".

4. Use the calculated threshold to create a binary DSM matrix DSM, with entries $DSM(p,q)$ that are either 1 indicating linkage or 0 indicating no linkage.

2. EXPERIMENTS

Two experiments were conducted with the new system:

1. **Accuracy and Specificity:** is the proposed method capable of identifying linkage in a given predictor-target set with a satisfactory degree of accuracy?

2. **Sensitivity:** how sensitive is the method to changes in both the problem (time series length, number of predictors) and the algorithm parameters (number of quantisation groups)?

Synthetic data was constructed for these experiments, using a fitness function for which the linkage was predetermined. The target time series was therefore computed directly, using the predictors and fitness function. The DSM output by the algorithm was then compared to the "true" DSM, and sensitivity/specificity measures were calculated.

The experiment structure and datasets are essentially the same for the two experiments. The fitness function was chosen to be the simplest possible while still displaying the necessary linkage: it is the sum of all the products of linked genes, i.e. $f(x) = \sum_i \prod_j x_{ij}$ where x is a chromosome and x_{ij} is the j-th gene in the i-th linkage group of that chromosome. The predictor functions are uniformly distributed random variables in the range [0,1].

be placed in one group while 0.90 would be placed in another. However, a better grouping could arguably have been to place 0.80 in one group and 0.89, 0.90 in the other, depending on the problem being studied.

2.1 Accuracy and Specificity

For the first experiment, we ran the algorithm on a number of datasets created as explained above, with the problem parameters held constant. The number of iterations was meant to mitigate the effects of any randomly occurring beneficial datasets. After a thousand runs on a 6-predictor dataset using a 5-digit alphabet for quantisation and a time series length of 500 observations, the algorithm showed a 97.7% percentage of True Positive (TP) results, along with an 81.8% percentage of True Negative (TN) results. The overall success ratio of the algorithm in these sets was 85% correctly identified DSM elements. The actual figures on the 15,000 gene pairs tested are given in Table 1.

Table 1: Accuracy and Specificity of the technique

	Linkage	No Linkage
Positive result	2930 (97.7%)	2181 (18.2%)
Negative result	70 (2.3%)	9819 (81.8%)

2.2 Sensitivity

The second experiment comprises a set of tests to measure the effects of different problem or algorithm parameters on the success ratio of our method. Each time all parameters except one are kept constant; the algorithm is run a number of times while changing that one parameter. In the case of altering problem parameters, since it is impossible to use the same dataset, the algorithm was run a number of times to mitigate the effects of randomly occurring beneficial datasets which might skew the results. In the case of algorithm parameters the same dataset is used for all the values of the variable being tested; this is then repeated for another dataset, and so on for (say) twenty repetitions. This should remove any anomalous effects of a specific dataset.

Three experiments were used to test parameter sensitivity: two tested sensitivity to the problem (number of predictors, and time series length) and the third tested sensitivity to a key aspect of the algorithm (quantisation alphabet length).

Number of predictors

Difficulties arise from a dataset with an increasing number of predictors. As the number of predictors increases, the noise in each linkage calculation is increased. This makes it more difficult to identify the true DSM. In the tests, the algorithm was run on datasets of 500 observations and n predictors, using values of n from 4 to 20 with a step of 2. The quantisation alphabet length was 5. There were 20 runs for each value of n and the results were averaged to reduce the effects of outliers. The average measures are plotted in Figure 4 and show an initial deterioration that starts to flatten once the number of predictors exceeds 12. At all times the TP rate is better than the TN rate.

Time series length

Increasing the time series length means increasing the potential for the appearance of suitable quadlets which will be used for the calculation of linkage measures. The more of these we are able to calculate, the more the effects of noise will be reduced, and the more accurate the DSM will be. We therefore explored the sensitivity of the algorithm to time series length using lengths from 100 to 1,000 with steps of

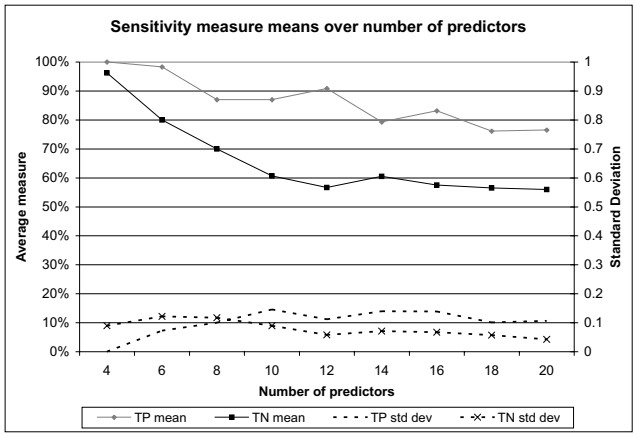


Figure 4: As the number of predictor series increases, linkage discovery becomes more difficult.

100. The alphabet length was 5 and the datasets consisted of 10 predictor time series and the target. At each length increment, the algorithm was run on twenty datasets and the specificity/sensitivity measures were averaged. The plot of these averages over the time series length clearly shows how the algorithm's performance improves as the population of chromosomes increases — see Figure 5.

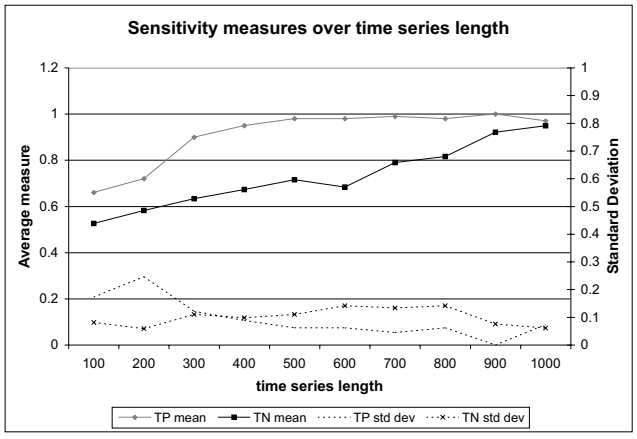


Figure 5: As the time series length increases, linkage discovery becomes easier.

Alphabet length

The only quantitative parameter of the algorithm is the alphabet size used in the quantisation. It can be shown that the magnitude of the effect that the alphabet size has on performance is problem specific. For example, an integer dataset with values from 1 to 5 would be completely unaffected by increasing alphabet size above 5. In a real-valued dataset, too many quantisation groups might split the available time points in such a way that not enough linkage measures are calculated, making the results sensitive to outliers.

We explored the sensitivity to alphabet length for our standard problem (see above). Specifically, we used 6 predictors of 500 observations with uniformly distributed real values in the range $[0, 1]$ and the target series was created using the function $target = P1*P2+P3*P4+P5*P6$. The alphabet length was varied from 5 to 15 with a step of 1. There were 20 runs for each increment of alphabet length, and the results were averaged. The plot of the averages over alphabet length is given in Figure 6.

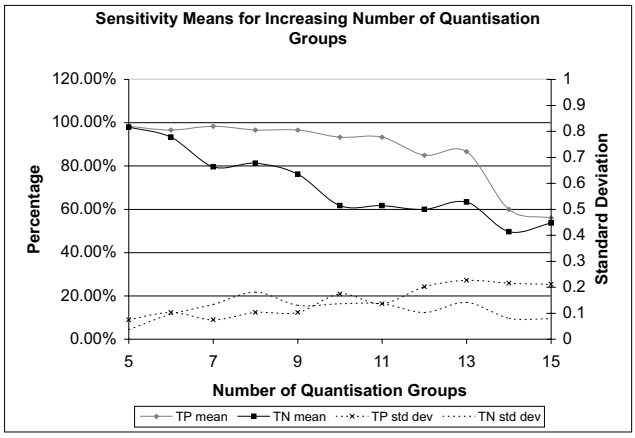


Figure 6: The effect of quantisation alphabet length (number of quantisation groups) is problem-specific: in this case increasing the alphabet length makes linkage discovery more difficult.

3. DISCUSSION

This section comprises three parts: a discussion of the results of the experiments; a discussion of an issue that arises in generating the binary DSM; and further work.

3.1 Discussion of results

On simple cases with few predictors, the algorithm was able to fully identify the linkage situation of a given dataset. The performance of the algorithm predictably deteriorated as the dataset length was reduced and/or as the number of predictors was increased. The alphabet size used in the quantisation process also affected the algorithm's ability to correctly identify linkage. These changes in performance are consistent across multiple runs of the algorithm on similar datasets; one positive aspect of this is that the algorithm's success is not merely the result of random factors. Yet how are we to interpret these results? Are they good or bad?

Discovering linkage in real-valued datasets is a difficult problem, and we should not expect 100% accuracy. But what constitutes an acceptable level of accuracy? We might, arbitrarily and subjectively, decide to aim for 90% success or better. But 90% of what measure? This depends on the context in which the linkage learner is being used. For the purposes of using linkage information to optimise the performance of a GA or GP, there are two cases to consider:

1. if we identify linkage where it does not exist, the consequence will be that the GA/GP misses an opportunity for full optimisation — it will not perform as quickly as perhaps it could, but at least it is still able to find an approximation to the globally optimal solution;

2. if we identify linearity where there is none (i.e. if existing linkage is not detected), the consequence will be that the GA/GP will attempt an optimisation that is

not well founded, and there is a possibility that the GA/GP will no longer be able to find an approximation to the globally optimal solution.

The first of these cases corresponds to the False Positive rate: the second corresponds to the False Negative rate. If we assume that it is much more important to find a better solution than it is to optimise performance, then the False Negative rate is the most important factor to consider.

This is good news for our system! Table 1 gives a False Negative Rate for our system of 2.3% which is impressive. However, Figure 4 shows that the False Negative Rate rises to 10% or more[4] for 8 or more predictors, which is bad.

Taking the results of all our experiments as a whole, it is clear that to get acceptable performance from the system (i.e. a False Negative Rate of 10% or less) it is necessary to ensure that the length of the time series is appropriate for the number of predictors — if there are more predictors, the time series need to be longer in order to get acceptable performance. This is plotted quantitatively in Figure 7.

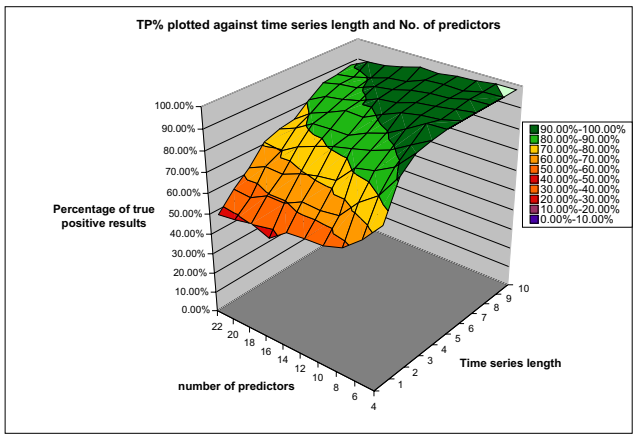


Figure 7: Increasing time series length can retain acceptable performance for larger numbers of predictors.

In summary therefore, we have shown that it is possible to obtain acceptable performance from the new algorithm as long as time series of appropriate length are available for higher numbers of predictors.

3.2 Thresholding

The binary DSM is constructed by dynamic determination of an appropriate threshold for the real-valued DSM. There is an assumption that pairs of predictors that are non-linearly linked will not have a low linkage value, and pairs that are not linked will not have a high linkage value.

The DSMDGA used a 2-means algorithm to find the appropriate threshold,[5] and we apply the same technique since it is straightforward to implement and relatively efficient. However, there are two possible variants:

1. Apply the 2-means algorithm to the values held in the real-valued DSM (each item DSM_{pq} is the mean of the linkage measures calculated for all occurrences of the schema that corresponds to the predictors p and q).
2. Apply the 2-means algorithm to the raw linkage values that were used to create the averages that are stored in the real-valued DSM.

Using the averaged linkage measures renders the algorithm unable to deal with extreme cases, i.e. cases where there is no linkage, or where the entire set of predictors is linked. This happens because the algorithm always separates the available measures into two groups, even though they might all belong to the same group. Consequently there will always be a non-zero number of ones and zeros in the binary DSM constructed using this method. This is not a problem in most cases since it would be rare for a problem to display either complete linkage or complete lack of linkage.

By contrast, when using all available linkage measures, the performance of the thresholding algorithm can be highly influenced by outliers. If there is a single value that is extremely high, it will drive the threshold upwards, possibly enough that other values which actually indicate nonlinearity would be below the threshold. This is a common occurrence, since in many problems a small number of linkage measures can be influenced more by the fluctuations of irrelevant predictors (i.e. the ones not part of the current nonlinearity check) than the predictors actually being tested. A telling example of this is given in Figure 8.

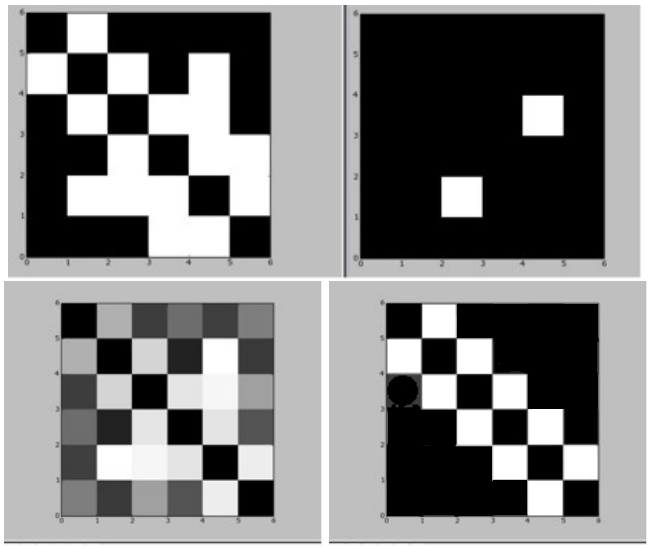


Figure 8: 2-means thresholding with averaged linkage measures (top left) and raw linkage measures (top right). Both were derived from the same underlying linkage data; the original real-valued DSM for this test problem is at the bottom left, and the ideal binary DSM for this test problem is at the bottom right. The threshold calculated using averaged linkage (left) is considerably better than that calculated using all the raw data (right).

3.3 Further Work

During development, certain modifications and extensions were considered, but could not be implemented due to time constraints. In our opinion, these represent valid subjects for further research, which could lead to substantial improvement in the algorithm's performance.

Different Quantisation Techniques. Preliminary experimentation with variations in the techniques used to convert

[4]The $FalseNegativeRate = 1 - TruePositiveRate$

[5]The DSMDGA must do this; not because it deals with real-valued predictors, but because it creates real valued averages of real-valued fitness.

the real-valued chromosomes to a discrete alphabet showed this can influence results. The intuition is that more sophisticated clustering algorithms will be better able to identify clusters and therefore reduce noise in the linkage calculation — we speculate that this improvement would be uniform across all cases, improving all true positive results and leading to a flatter surface for Figure 7. Since these effects are problem-specific, it will be important to observe the algorithm's behaviour on more structurally diverse datasets.

Different Thresholding Techniques. A key feature of the method proposed in this paper is the calculation of the threshold. Alternate clustering methods could be used in order to produce more sophisticated thresholds. While this was unnecessary in previous applications of dependency structure matrices, it is important to calculate the best possible threshold for use by this method since the algorithm must produce a final result on its first run.

More sophisticated problems. The eventual aim of this work is to identify the linkage of real financial time series in a way that is beneficial to the performance of a GA/GP application. It is important, therefore, to test the algorithm's performance in (i) problems more closely approximating actual time series; and (ii) real-world financial time series where the underlying linkage is understood. An example of the former would be the use of predictors following the Gaussian distribution, or even predictors constructed as Martingales, i.e. as a Markov process with zero-mean shocks. The results of these experiments might also better indicate which aspects of the algorithm would, if refined, improve performance the most.

Different problem domains. Though this work was motivated by our desire to analyse financial time series data, it is not restricted to that problem domain. Our extension to the DSMDGA is applicable to any situation where the underlying domain is a finite set of discrete real-valued observations. It would therefore be of interest to extend our work to the consideration of other suitable domains.

4. SUMMARY AND CONCLUSIONS

Standard linkage-learning algorithms fail for typical GA or GP analysis of financial time series. The problem extends to any situation where linkage must be detected in series of discrete real-valued observations. We have demonstrated that perturbation techniques cannot be used, and the DSMDGA as it stands cannot be used. We therefore introduced a new algorithm, an extension of the DSMDGA, to learn linkage directly from the observed real values.

We have assessed the accuracy of the new algorithm and provided a characterisation of its sensitivity to the number of predictors and the dataset length. We propose that the most important accuracy metric for linkage learning in our context is the False Negative Rate, which we further propose should be 10% or less. We have shown how this accuracy can be achieved or bettered by the new algorithm as long as the length of the time series dataset is appropriate for the number of predictors. Figure 7 suggests that for ten or fewer predictors the required dataset length is approximately $100 + NumberOfPredictors * 50$, and for more predictors the required length is $NumberOfPredictors * 50$.

Financial time series vary in length according to whether the data is observed monthly, daily or on a "per-tick" basis. A monthly dataset for twenty years would have 240 observations, which would be sufficient to support accurate linkage analysis of up to 4 predictors. By contrast, daily data for five years would have just over 1,800 observations, sufficient to support accurate linkage analysis of 36 predictors.

We conclude that the new algorithm exhibits good behaviour within practical limits and is suitable for determining non-linearity linkage to support GA or GP analysis of financial time series data.

5. ACKNOWLEDGMENTS

The authors thank David Coffin at UCL for his assistance in the early stages of this work, and the four anonymous referees for their helpful comments.

6. REFERENCES

[1] D. E. Goldberg and M. Munetomo. Designing a Genetic Algorithm Using the Linkage Identification by Nonlinearity Check. IlliGAL Report 98014, Illinois Genetic Algorithm Laboratory, 1998.

[2] R. B. Heckendorn and A. H. Wright Efficient Linkage Discovery by Limited Probing In *Evolutionary Computation*, **12**(4):517–545. MIT Press, 2004.

[3] J-W. Li and M-Q. Li The study of epistasis based on the random walk model in fitness landscapes of schemata. In *Proc. Conf. on Machine Learning and Cybernetics*, Vol. 3, pp. 1396–1400. IEEE, 2002.

[4] J. B. MacQueen Some Methods for classification and Analysis of Multivariate Observations In *Proc. Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1, pp. 281–297. University of California Press, 1967.

[5] M. Munetomo and D. E. Goldberg. Identifying linkage by Nonlinearity Check, IlliGAL Report 98012, Illinois Genetic Algorithm Laboratory, 1998.

[6] M. Tezuka, M. Munetomo, and K. Akama. Linkage Identification by Nonlinearity Check for Real-Coded Genetic Algorithms. In *Proc. Genetic and Evolutionary Computation Conference (GECCO 2004)*, LNCS **3103**: 222–233. Springer-Verlag, 2004.

[7] S. Tsutsui, D. E. Goldberg, and K. Sastry. Linkage Learning in Real-Coded GAs with Simplex Crossover. In *Proc. Conf. on Artificial Evolution*, pp. 73–84, 2001.

[8] S. Tsutsui and D. E. Goldberg. Simplex Crossover and Linkage Identification: Single-Stage Evolution VS. Multi-Stage Evolution. In *Proc. Congress on Evolutionary Computation*, pp. 974–979, 2002.

[9] T-L. Yu and D. E. Goldberg. Conquering hierarchical difficulty by explicit chunking: Substructural chromosome compression IlliGAL Technical Report 2006007. Univ. of Illinois at Urbana-Champaign. 2006.

[10] T-L. Yu, D. E. Goldberg, A. Yassine, and Y-P. Chen. A Genetic Algorithm Design Inspired by Organizational Theory: A Pilot Study of a Dependency Structure Matrix Driven Genetic Algorithm. IlliGAL Technical Report 2003007. Univ. of Illinois at Urbana-Champaign. 2003.

[11] T-L. Yu, D. E. Goldberg, A. Yassine, and Y-P. Chen. A Genetic Algorithm Design Inspired by Organizational Theory: Pilot Study of a Dependency Structure Matrix Driven Genetic Algorithm. In *Proc. Genetic and Evolutionary Computation Conference (GECCO 2003)*, LNCS **2724**:1620–1621. Springer-Verlag, 2004.