Stages of Gene Regulatory Network Inference: the Evolutionary Algorithm Role

Alina Sîrbu, Heather J. Ruskin and Martin Crane
Centre for Scientific Computing and Complex Systems Modelling
School of Computing, Dublin City University, Dublin 9
Ireland

1. Introduction

Uncovering interactions between genes and their products has been a major aim of Systems Biology over recent years, (Przytycka et al. (2010) and references therein). The objective is to gain a better understanding of the functioning of different organisms, together with discovery of disease markers and new treatments, (Bar-Joseph, 2004; Tan et al., 2008). Gene regulatory network (GRN) analysis has been facilitated by the advent of technologies for measuring gene expression: these include mature technologies such as *qrtPCR*, (Logan et al., 2007), suitable for a limited number of genes, and *microarrays*, (e.g. Baldi & Hatfield (2002)), which allow for high-throughput measurement of thousands of genes at the same time. More recently, *RNA-Seq* (Hurd & Nelson, 2009) measurements have become available, due to advances in high throughput *sequencing* technology, but these data are still scarce, due to high experimental costs. Characterised as they are by high dimensionality and noise levels, analysis of these data is far from trivial. The class of computational methods known as Evolutionary Algorithms, (EAs), has demonstrated relevance for different investigative targets, (Pal et al., 2006; Sîrbu et al., 2010a). This chapter, in consequence, presents an overview of approaches and issues in GRN modelling and inference, and discusses the role of EAs in this regard.
Three different analysis stages can be identified for GRN inference: (i) expression pattern analysis, (ii) mathematical modelling from expression data and (iii) integrative modelling. At each of these, and most particularly at the last stage, EAs have an important role to play, due to the strength and flexibility of these search methods.
Expression pattern analysis is largely concerned with the application of classification and clustering methods to gene expression data. Clustering of genes, as a first step towards GRN modelling, (Lee & Yang, 2008; Thieffry, 1999), together with classification, which aims at assigning samples to different classes (usually for gene expression data, to distinguish between tissue types, e.g. control/treatment or healthy/infected, for diagnostic purposes), give valuable insight on gene involvement in different processes. EAs are typically employed at this stage, with some success, for feature selection and clustering.
At the second stage, a GRN model is created to explain the data (see e.g. He et al. (2009) for a review), which can be used for *in silico* simulation and process analysis under various criteria. Such a model is built by reverse engineering from available time course expression data, with inferential algorithms used to fit model parameters to the data, using evolutionary optimisation. Different EA approaches are presented here, for inferences on *discrete qualitative*

to *continuous quantitative* models. Consequently, the discussion includes *classical* to *hybrid* EAs, and identification of strengths and weaknesses.

A general limitation in GRN modelling is that, although qualitative models can be built for entire GRNs, quantitative analysis is still restricted to sub-networks, due to limited data available and the large number of parameters to be optimised. Quantitative models allow for a better representation of interaction links, and for continuous simulation of dynamical behaviour, but the limitations in size and accuracy have impeded their use in real-world scenarios. Consequently, a third stage in network inference, integrative analysis, (Hecker et al., 2009), aims at reconciling different sources for the large amount of biological data available, in order to improve reliability of the inferential process, and realism of the models. This is not without risk, as multi-source data can contain heterogeneous noise, which has to be dealt with. Further, large scale integrative analysis requires a large amount of computational resources, and algorithms have to be optimised and parallelised to address this. Additional data types, which can contribute to this synthesis, include DNA-protein interactions, knock-out/knockdown experiments, binding site affinities, as well as known transcription factors (TFs) and RNA interference measures.

To date, integration efforts are sparse. Nevertheless, examples of approaches based on EAs are presented here, although these typically combine only *one* additional data type with expression measurements. Ideally, all related data should contribute to the inferential process. With this aim, a novel algorithm, based on evolutionary computation, that aims at large scale data integration for quantitative modelling, is also outlined, and the advantages and disadvantages of EAs for data unification discussed.

The rest of this chapter gives background on GRNs, general modelling methodology and mathematical models in Section 2, then follows the development of existing evolutionary algorithm approaches through the three stages of inference in Sections 3, 4 and 5. Section 5 also describes a novel approach for data integration, which builds a framework for inclusion of multiple data types in the inferential process, followed by a concluding discussion on EA role in Section 6.

2. Background

2.1 Gene regulatory networks (GRNs)

DNA encodes the information the cell needs to create proteins that are vital for its mechanisms (e.g. Brown (2002)). Each cell of an organism contains the same information, i.e. the DNA sequence, but in different tissues, cells will behave differently. This indicates that other mechanisms must exist to control protein levels depending on the environment; one such example is the gene regulatory network.

The Central Dogma of Molecular Biology describes gene expression[1] as $DNA \rightarrow RNA \rightarrow$ protein, but, in fact, the process is more complex as it consists of several stages and can be influenced by several factors at each stage, (Brown, 2002). The *initiation of transcription* is one of these stages, influenced by proteins called transcription factors (TFs). These TFs bind to the region upstream of the gene that needs to be expressed and regulate its transcription in a positive or negative way, (up- or down-regulation). Such interactions create a regulatory network between TFs and genes, i.e. a GRN. As TFs are in turn encoded by other genes, we can consider these interactions as being between pairs of genes instead of gene-protein pairs. GRNs are used to control protein levels during biological processes, and perturbations in the network lead to unwanted behaviour, i.e. disease.

[1] Formation of a protein from the corresponding gene.

GRNs feature a set of characteristics that distinguish them from random networks (Marbach et al., 2009). They are scale-free, modular and contain *network motifs*, i.e. patterns of interactions that appear with high frequency and control oscillatory behaviour of the network. Also, the in-degree of the nodes, (i.e. the number of regulators for each gene), is bounded by a small number compared to the total number of genes in the network. These properties are very important, as they can be used to enhance the inferential process and also to generate plausible *synthetic networks* which are used to validate inferential algorithms.

Several technologies that measure gene expression have been developed, typically concerned with gene activity at the mRNA[2] level. Among these, high-throughput technologies, (microarrays, Baldi & Hatfield (2002), and RNA-Seq, Hurd & Nelson (2009)), allow for measurement of mRNA concentrations for a large number of genes at the same time. These measurements can be viewed as snapshots of the expression levels of genes under certain conditions and, with a large-enough set of snapshots, it is theoretically possible to uncover the underlying GRN (Liang et al., 1998).

2.2 From gene expression data to GRNs

Gene expression data consists of the expression levels of many genes under multiple conditions, (Stekel, 2003). Hence, for each gene, a vector of values shows the *gene expression pattern* for a number of different experiments, (Figure 1a). At the same time, the data can be viewed as a set of vectors describing the behaviour of the organism under certain conditions, (experiments), i.e. the *experimental patterns*, which represent expression values for many genes in a single experiment, (Figure 1b). By analysing both pattern types, (separately or together), useful knowledge related to the connections between genes or the similarity between conditions can be found. The ultimate aim is to build a reliable model for the underlying GRN, which can be further used for *in-silico* simulation and analysis of the system. This goal has triggered significant research efforts, (e.g. He et al. (2009) and references therein), which can be classified, based on the type of analysis performed, specifically: (i) expression pattern analysis, (ii) modelling from time series data and (iii) integrative modelling. Evolutionary algorithms have had an important role in the different stages of analysis, due to their flexibility and search power.

The *first stage* in studying gene expression data for GRN discovery applies pattern analysis algorithms on both experimental and gene patterns seen in the data. Such algorithms include clustering, classification and feature selection techniques.

Clustering is considered here as unsupervised [3] learning where a set of data entries has to be grouped into clusters, based on their attribute values, (Manning & Schütze, 1999). The clusters and cluster assignment for the training data set are typically not known beforehand, but are deduced based on dissimilarity or distance measures. Such measures may be statistical constructs, such as correlation, as well as standard spatial distance measures: Euclidean, Manhattan and others. Bi-clustering is also a variant that has been widely applied to gene expression data, (Kerr et al., 2008). This aims at grouping both genes and experiments at the same time, indicating not only clusters of co-expressed genes, but also in which experiments these appear.

[2] Messenger RNA (mRNA) is the RNA that results from transcription of DNA during the expression of a gene. mRNAs are translated into proteins based on their nucleotide sequence. Other different types of RNA exist, but relate to other functions in the organism.

[3] Recently, supervised clustering methods have emerged from the need for more control over the meaning of the resulting clusters or the features that are considered by the unsupervised clustering technique. However, this section concentrates on unsupervised clustering.

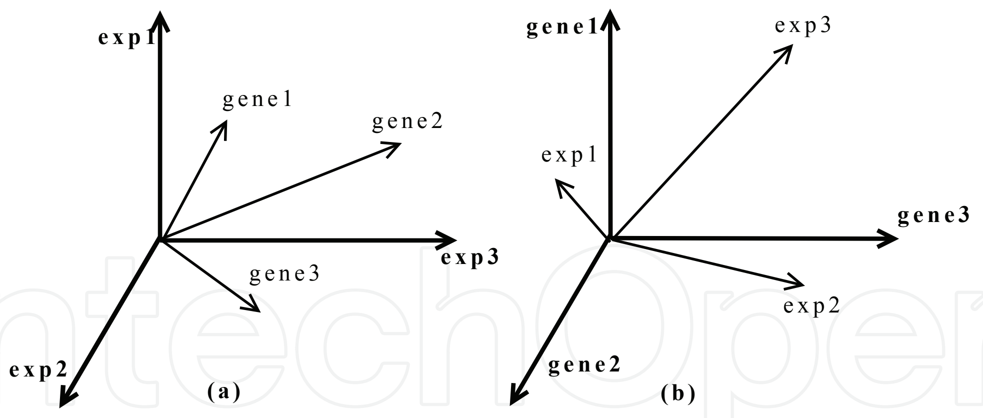


Fig. 1. Gene expression data interpretations: (a) a set of vectors representing expression values for one gene under different experiments and (b) a set of vectors representing expression values for multiple genes under a single experiment

In the case of gene expression data, clustering gives insight on the relationships between genes and, in consequence, is considered the *first step towards gene network inference* (Lee & Yang, 2008; Thieffry, 1999). Genes that belong to the same cluster are assumed to be co-regulated, (i.e. regulated by the same protein complex), or co-regulating, (i.e. regulating each other). Once the clusters are generated, the objective is to look for binding site motifs in the precursors[4] of the genes in each cluster. In this way it is possible to find unknown binding sites or to generate hypotheses on which proteins regulate the co-regulated genes in the cluster, (based on previous knowledge on regulatory motifs). These hypotheses can be further validated by laboratory experiment.

A gene expression dataset can contain thousands of genes so that the elements to be clustered/classified are points in a high-dimensional space, hence analysis is computationally intensive. Also, as data are intrinsically noisy, the high number of dimensions can bias the algorithm convergence. It is possible that some features of the gene expression data are redundant, (Liu et al., 2002), hence the need to develop feature selection techniques, in order to make analysis more efficient. Subsection 3.2 describes some of the feature selection methods, which have been applied in the context of clustering or classification of microarray data. Although classification of different experiments, (for diagnostic purposes, such as distinguishing between infected and healthy tissue), may not have an immediate use in GRN inference, feature selection techniques do give an indication as to which genes are most important in the processes under analysis, so we have included them at the first inferential stage.

A *second stage* in GRN inference is mathematical modelling using time series gene expression data. In these data, gene expression levels are measured over time, with each experiment in the data describing a different time point. These series patterns can be modelled using mathematical tools, of which a large number have been applied to GRNs, (see e.g. He et al. (2009); Lee & Tzou (2009) and references therein). Generally, the process of modelling GRNs consists of a few main steps: choosing an appropriate model, inferring parameters from data, validating the model and conducting simulations of the GRN to predict its behaviour under different conditions. Due to the large number of genes in such datasets, clustering methods (stage one) have been applied by some authors for dimensionality reduction (either by considering cluster centroids as being one gene in the network, Wahde & Hertz (2000), or by analysing subsets of genes corresponding to selected clusters, Lee & Yang (2008)).

[4] The region in the DNA sequence located before the gene.

In order to model a GRN, genes are taken to be variables that change their (expression) values over time. Depending on variable type, methods can be classified as discrete or continuous, deterministic or stochastic, or as *hybrid* (using more than one type of variable). Two broad approaches are described in the literature (Lee & Tzou (2009)): coarse-grained and fine-grained models, with the former containing less detail on the interactions between genes. Usually, coarse-grained models use discrete variables, while fine grained models use continuous ones. A GRN can be very large and can contain complicated interactions, so a fine-grained model will have an enormous number of parameters to deal with. Analysis of this kind of model is very complex, so viewing the network 'top-down', in order to be able to analyse it globally, is the aim of coarse-grained models, (e.g. Linden & Bhaya (2007); Maki et al. (2001); Repsilber et al. (2002)). Other authors, (e.g. Kikuchi et al. (2003); Morishita et al. (2003); Noman & Iba (2006); Tominaga et al. (1999); Wahde & Hertz (2000); Xu et al. (2007)), have chosen to focus on detailed models, but for analysis of sub-networks only of the entire GRN. A useful approach, clearly, is to combine the two levels of detail, moving between the coarse and fine-grained model to highlight key biological knowledge (Maki et al., 2001).

To reverse engineer GRNs, EAs require a specified model type and data set. This enables parameter evolution to be monitored and performance in terms of fitting input data to be evaluated. A population of such parameters representing different models, (also known as population of *candidate solutions* or *individuals*), evolves towards a better set, by applying genetic operators (e.g. crossover and mutation). The fitness function is typically defined as the difference between the observed data and the output of the model, (squared, or averaged over the data points), as described in Equation 1.

$$\text{fitness} = \sum_{i=1}^{n} \sum_{t=1}^{T} (x_i(t) - y_i(t))^2 \tag{1}$$

where $x_i(t)$ is the expression value of gene i at time t, observed in real experiments, and $y_i(t)$ is the expression value of gene i at time t generated by the model. Since every model has its distinctive features, steps in the algorithm differ from one approach to another, but the main skeleton is usually preserved. In this chapter we describe several such methods, following the development from classical to advanced hybrid methods.

The ideal model for a GRN would be fine-grained, accounting for all the features of the real GRN, applied to the entire genome in a cell. Achieving such a model is a non-trivial task, as most methods to date are either too coarse or can not model large systems. Also, existing gene expression time-series data are insufficient to infer the large number of parameters for such a detailed model, resulting in an *under-determined* problem. However, a very large pool of biological knowledge, from different types of experiments, does exist in the literature. The issue then is whether it is possible to combine all existing knowledge in an attempt to improve system inference. Approaches that use gene promoter data, results from protein-protein interaction experiments, knock-out microarray experiments and other related information have started to appear, (Hecker et al. (2009) and references therein), opening the road for a *third stage* of GRN inference, based on *heterogeneous data integration*. Some efforts have used evolutionary computation but are at an early stage only, so that they benefit only partially, at best, from the flexibility offered by EAs for large scale data integration. We aim to address this question also, (Section 5).

In the rest of this section, an outline of mathematical models, used in conjunction with evolutionary algorithms for GRN modelling, is provided.

2.2.1 GRN modelling approaches

Boolean networks

Boolean networks are coarse-grained models for GRNs that use Boolean values for gene expression: the gene is on/off with values 1/0 respectively (Liang et al. (1998)). Regulation is expressed in terms of Boolean functions attached to each gene :

$$Y_i = F_i(X_{i_1}, .., X_{i_k})$$

where $X_{i_1}, .., X_{i_k}$ are the binary expression levels of regulators of gene i and Y_i is the predicted expression value for gene i. This model is very well suited to modelling large networks, as it does not require a large number of parameters. Due to their relative simplicity and limited detail, Boolean networks have been employed in the analysis of steady states and general behaviour of GRNs. However, they have a few disadvantages as they can not simulate continuous behaviour and complex nonlinear interactions, characteristic of GRNs. Additionally, discretisation of expression values, which are continuous, may lead to information loss, which can result in fewer interactions identified.
A generalisation of the Boolean network is the multistate discrete network (Repsilber et al., 2002). In this model, gene expression levels can take more than two discrete values (in the set $S = \{0, .., n\}$) and the transition functions are general functions $F_i : \{0, .., n\}^k \rightarrow \{0, .., n\}$, mapping between current expression values for all genes and that of gene i at the next time point.

Rule sets

Another model of regulation uses different types of rules to explain the observed patterns in the data. This approach has the advantage of being more intuitive, as relationships between genes are expressed using natural language. One such model uses fuzzy rules, (Linden & Bhaya, 2007), which are based on the notion of fuzzy sets. These sets have imprecise boundaries, defined by a membership function: applied to any element in the universe, they return a number in the interval [0,1], representing the degree to which that element is a member of the current set. A fuzzy rule is a conditional of the form *if x is in A then y is in B*, which specifies a relation between fuzzy sets *A* and *B*. Every fuzzy rule also has a membership function that specifies the degree of truth of the implication.

Ordinary differential equations

The models described in previous paragraphs are coarse-grained models. These use discrete states for gene expression values, with the influences of a given set of genes on other genes described qualitatively, rather than quantitatively. However, gene interactions are very complex and, in order to model these, a fine-grained continuous model is needed, which considers interactions quantitatively. One such model is a system of differential equations.
Ordinary differential equation systems express the change in the expression level of each gene in time as a function of the expression levels of other genes, but make no other assumption about the mathematical form:

$$\frac{dx_i}{dt} = F_i(x_1, .., x_n) \tag{2}$$

where x_i represents the expression level of gene i. The inferential algorithm, therefore, is not restricted to a prescribed set of functions and can model complex behaviour. At the same time, few constraints mean that the search space is very large and more sophisticated methods are typically required to refine the analysis.

Linear differential equations

The simplest model example, and one that has received a lot of attention (e.g. (Akutsu et al., 2000; Ando & Iba, 2003; Deng et al., 2005)), is the linear system of differential equations. This simplifies the way genes interact, by using linear dependencies but at the same time retains the continuous aspects inherent in differential equations. This simplification results in loss of modelling power, compared to a nonlinear model choice, (as gene interactions are known to be more complex), but gains from the perspective of simpler inference.
This model describes changes in gene expression values as:

$$\frac{dx_i}{dt} = \sum_{j=1}^{n} w_{ij} x_j \tag{3}$$

where x_i and x_j represent expression values of genes i and j and w_{ij} the regulation *strength* of gene j on gene i. A negative value for w_{ij} corresponds to repression of gene i by gene j, a positive value corresponds to activation and a null value to no effect of gene j on gene i. Different versions of the model exist, which add other terms to the equation, accounting for external stimuli, degradation rates or noise. The system can be described by the matrix $\mathbf{W} = (w_{ij})$, also known as the *interaction or regulation matrix*. Inferring a model means finding the values in $\mathbf{W}$.

S-Systems

Although linear systems improve the level of detail achieved by the modelling approach, these still exclude some information. Regulatory networks are intrinsically *nonlinear* systems and approaches that correctly model gene interactions are needed. S-Systems are a special type of differential equation systems, based on power-law formalism, and are capable of capturing complex dynamics. The disadvantages are an increase in the number of parameters and reduction in the available choices of reverse engineering techniques, as linear regression methods are not applicable any more. The equations in S-Systems are of the form:

$$\frac{dx_i}{dt} = \alpha_i \prod_{j=1}^{n} x_j^{g_{ij}} - \beta_i \prod_{j=1}^{n} x_j^{h_{ij}} \tag{4}$$

The two terms correspond, respectively, to synthesis and degradation influence from other genes in the network; specifically, α_i and β_i, are *rate constants* and represent basal synthesis and degradation rate, while g_{ij} and h_{ij}, (*kinetic orders*), indicate the influence of gene j on the synthesis and degradation of the product of gene i.

Linear time-variant model

The linear time-variant model expresses the regulatory effect on a gene using a linear expression:

$$r_i(t) = \sum_{j=1}^{n} w_{ij}(t) x_j \tag{5}$$

and computes the expression value of that gene at time $t+1$ by applying a sigmoid function [5] to this effect:

$$x_i(t+1) = \frac{1}{1+\exp(-r_i(t))} \tag{6}$$

[5] Mathematical function that has an S-shaped graphical representation. One example of a sigmoid function is the logistic function that restricts the output to the interval$(0,1)$: $f(x) = \frac{1}{1+e^{-x}}$

The values $w_{ij}(t)$ represent the interactions between genes, and are expressed using a Fourier series, as described in Equation 7.

$$w_{ij}(t) = \alpha_{ij} sin(\omega_i t + \phi_{ij}) + \beta_{ij} \tag{7}$$

Thus, interactions are modelled by a linear term, β_{ij} and a non-linear sinusoidal term.

Partial differential equations

The differential equations models, presented so far, do not take into account the spatial distribution of cells and gene products. However, in certain situations, such as cell differentiation during development, the spatial information is very important, so more complex differential equation-based models are needed (partial differential equation systems, introduced by Baldi & Hatfield (2002)). These express concentration changes in both space and time, by reaction-diffusion equations. Here, the one-dimensional version of these equations is described, but these can be extended to 2 or 3-D situations. Considering a linear sequence of L compartments or cells, the concentration of product i in cell l depends on the regulatory effects in cell l but also on the diffusion process between this cell and its neighbours. Diffusion is considered to be proportional to the concentration difference between the two cells. So, the differential equation that describes this process is:

$$\frac{dX_i^{(l)}}{dt} = F_i(X_1^{(l)}, .., X_n^{(l)}) + D_i(X_i^{(l-1)} - 2X_i^{(l)} + X_i^{(l+1)}) \tag{8}$$

where F_i are the regulation functions and D_i are diffusion functions. This is for the case when space is discrete (well delimited cells and compartments). In the continuous case, the concentrations of the products are functions of both time and space, so the system can be modelled with equations of the form

$$\frac{\partial X_i}{\partial t} = F_i(X_1, .., X_n) + D_i(\frac{\partial^2 X_i}{\partial s^2}) \tag{9}$$

where s is the space variable.

Artificial neural network models

ANNs are very suited for modelling complex behaviour as, any function can be simulated, by adjusting the weights. A type of ANN that has been repeatedly used to model GRNs is the recurrent neural network (RNN) (Lee & Yang, 2008; Vohradsky, 2001; Wahde & Hertz, 2000), which model dependencies between genes as:

$$\frac{dx_i}{dt} = m_i S(\sum_{j=1}^{n} w_{ij} x_j + b_i) - r_i x_i \tag{10}$$

where r_i is the degradation rate of gene product i, b_i accounts for external input, m_i is the maximum expression rate and x_j are expression levels while S is a sigmoid function. This model is similar to that of linear systems of differential equations; however the introduction of the sigmoid function allows for modelling non-linear behaviour. A variant of this model considers discrete time points, and computes the expression value of gene i at time point $t+1$ using the values of the regulators at time t:

$$x_i(t) = S(\sum_{j=1}^{n} w_{ij} x_j + b_i - r_i x_i) \tag{11}$$

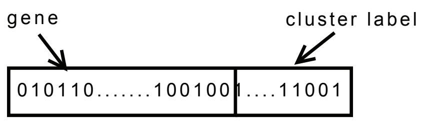


Fig. 2. Chromosome representation in GenClust

This reduces the computational cost for simulating the data, as no differential equations are involved, which is an important advantage in the context of evolutionary optimisation, (which requires simulation for every fitness evaluation).

3. Stage 1 : Pattern recognition methods

3.1 Clustering

Evolutionary algorithms have been applied in clustering gene expression data, either individually, (Di Gesu et al., 2005), or by hybridisation with classical clustering methods (Lu et al., 2004a;b). Additionally, evolutionary bi-clustering methods have been developed and applied to this type of data (Chakraborty & Maka, 2005; Mitra & Banka, 2006).
GenClust (Di Gesu et al., 2005) is a novel method, using a genetic algorithm-like approach. It differs from other genetic algorithms in that the population does not represent a set of possible solutions, but only one. Each individual in the population encodes one sample and a label representing its cluster (Figure 2). By analyzing these labels the components of each cluster can be computed. The approach incorporates elements of EC, such as genetic operators, that are applied at each generation. To generate a secondary population, classical one-point crossover and bit-flip mutation are applied to each individual with given probability. After applying the operators, redundancy and inconsistency have to be removed from the population, so that it still represents a partition of the data set. Fitness evaluation is based on the sum of intra-cluster variances. The aim of the algorithm is to minimise this measure and, consequently, to obtain tight clusters. The method has been validated on five datasets (Rat Nervous System, Reduced Yeast Cell Cycle, Yeast Cell Cycle, Peripheral Blood Monocytes and Reduced Peripheral Blood Monocytes) and compared with other clustering methods like K-Means. The algorithm has been shown to converge rapidly to a local minimum; however, the resulting clusters were comparable those obtained by other techniques.
A similar objective was pursued by Lu et al. (2004a;b), where a hybrid genetic K-Means algorithm (GKA) with two different versions (FGKA - fast GKA and IGKA - iterative GKA) was introduced. Hybridisation with K-Means consists of a custom genetic operator, based on this classical clustering method, which changes cluster allocation to the closest centroid in random individuals. This operator is applied to each individual with given probability. Neither FGKA nor IGKA uses a crossover operator, and mutation is performed based on dynamically computed probabilities depending on current cluster assignment. The difference between the two algorithms is that the latter updates cluster centroids and within cluster variance each time a mutation is performed on an individual, while the former computes these for each generation. This makes IGKA faster when mutation probabilities are low, while FGKA is faster when these are large. In consequence, a hybridisation of the two (HGKA-hybrid GKA) is also proposed (Lu et al., 2004b). The algorithms were applied to microarray yeast and serum data and IGKA was shown to obtain better clusters of genes from the same functional categories.
In Chakraborty & Maka (2005) a genetic bi-clustering algorithm based on K-Means and greedy local search seeding is presented. The algorithm was applied to yeast and human lymphoma data and was shown to provide better bi-clusters when validated against previous biological

knowledge, compared to Cheng & Church (2000) (which adopts a greedy search approach). A similar algorithm, that of Mitra & Banka (2006), employs multi-objective optimisation for bi-clustering. The algorithm is initialised using a greedy algorithm based on random initial solutions. Two objective functions are used, one maximising the number of genes and conditions in the bi-cluster, and another maximising homogeneity. Method evaluation was performed on the same yeast and human lymphoma datasets, and results indicate better performance compared to the single objective variant and to simulated annealing for bi-clustering Bryan (2005).

3.2 Feature selection techniques

Given the high dimensionality of gene expression data, clustering and classification are computationally expensive. However, a large fraction of the genes in these datasets are not differentially expressed between different experiments, (i.e. are redundant), so these could be eliminated from the analysis to reduce dimensionality. To achieve this, feature selection techniques have been formerly applied to gene expression data, mainly for classification purposes. Such methods select features (genes) that are important in the process under analysis, as they display a change in expression from one experiment to another. This filtering not only reduces the computational cost, but also improves pattern recognition for participating elements.

Feature selection methods can be classified into two categories: wrapper and filter methods. *Filter* methods compute for each feature a measure of relevance for the current classification task. The features are sorted by their relevance and the top n are further used for pattern recognition. *Wrapper* methods, on the other hand, use the classifier itself to find the importance of a set of genes. They select a feature subset and train a chosen method on that set. The performance of the trained classifier can be seen as a measure of the relevance of the genes in the subset. The wrapper method iterates this operation for different subsets and chooses the best one, and the difficulty is how to choose feature subsets that maximise the accuracy of the classifier, while minimising the number of selected genes and iterations. The search space for this problem is huge: if the number of initial genes is n, 2^n possible subsets exist. In this context, evolutionary techniques are known to cope well, as they benefit from mechanisms obtaining good solutions by searching a small portion only of the entire space, (Baeck et al., 2000). Consequently, there are several approaches that use EAs as wrapper methods for feature selection, for example Li (2001); Li et al. (2004); Ooi & Tan (2003); Shah & Kusiak (2004); Souza & Carvalho (2005). One of these has also been applied to proteomics data, (Li, 2001; Li et al., 2004).

Most evolutionary approaches for wrapper methods are very similar. A population of gene subsets is maintained and allowed to evolve using different genetic operators. The fitness of each candidate solution is a measure based on the training error of the classifier when using that specific set of features. After applying genetic operators, the fittest individuals remain in the next generation. While principles are the same, existing methods differ in terms of classifier used or EA components, (e.g. size of the chromosomes, fitness function, etc). Additionally, some methods ,(Li, 2001; Liu et al., 2009; Shah & Kusiak, 2004), aggregate features obtained in multiple runs in order to improve performance. Table 1 summarises existing EA wrapper methods.

Recently, a new method for feature selection using genetic algorithms has been developed (Zhu et al., 2007). This is a hybrid of the wrapper and filter methods: two operators that add or remove features from a set, in a filter-like manner, are applied to the feature set encoded by the best individual of each generation. In this way, the individuals of the genetic algorithm, (candidate feature sets), are fine tuned to improve the overall fitness and reduce the number of

Method	EA	Classi-fier	Multi-class	Feature set size	Combining results	Fitness	Datasets
Li (2001); Li et al. (2004)	GA	K-NN	No	Fixed	Filter by appearance count	Classifier accuracy	Leukemia (microarray), Ovarian cancer (SELDI-TOF)
Shah & Kusiak (2004)	GA	Decision tree	No	Fixed	Reunion or intersection	Classifier accuracy	Emulated
Ooi & Tan (2003)	GA	Bayesian	Yes	Variable	None	Classifier error rate in cross-validation and independent test	9 cancer types, 14 cancer types
Souza & Carvalho (2005)	GA	SVM	Yes	Variable	None	Classifier error rate and feature set size	Leukemia, Blue-cell tumour
Liu et al. (2009)	GA	ICA-SVM, P-ICR	No	Variable	Intersection	LOOCV + feature set size	Colon cancer, High-grade glioma

Table 1. Features of EA wrapper methods. Abbreviations: ICA - Independent Component Analysis, (Liu et al., 2009), GA- genetic algorithm, SVM - support vector machine, K-NN - K - nearest neighbours, LOOCV - leave one out cross validation

generations. The algorithm uses an SVM as a classifier. The two newly introduced operators are based on the *Markov Blanket* concept. The Markov Blanket of a feature F is the set M of features that satisfies :

$$P(\mathcal{F} - M - F|F, M) = P(\mathcal{F} - M - F|M), \tag{12}$$

where $\mathcal{F}$ is the set of all possible features. So, the probability of the values for all features except F and M are independent of F, given M. Intuitively, in our case, if the Markov Blanket of feature F is in a subset of features, then it can bring no more information to that subset so it can be removed. The algorithm was applied by the authors on 11 different datasets, including ones for lung or breast cancer, and compared to other filter and wrapper feature selection methods. It was shown to perform better than other methods for most datasets. In Zhu & Ong (2007), also, another hybrid filter-wrapper genetic algorithm-based feature selection algorithm is presented, which implements the new genetic operators using a ranking method, i.e. Robnik-Łikonja & Kononenko (2003), instead of the Markov Blanket. This is shown to have similar results in terms of accuracy. However, the Zhu et al. (2007) algorithm finds smaller gene sets, so it has an important advantage in terms of practical use: the smaller the number of features, the less expensive the diagnostic procedure. Further, the Markov Blanket embedded genetic algorithm has also been applied very recently to multi-class problems, (Zhu, Jia & Ji, 2010; Zhu, Ong & Zurada, 2010), using multi-objective optimisation, (where each objective corresponds to the accuracy of a bi-classification task in a one-versus-all manner). Here, the notions of *full class relevant* and *partial class relevant* features are introduced. These, respectively, are features that have a role in differentiating all classes (i.e. display different expression levels in all classes) and features that differentiate only part of the classes (i.e. may have similar values is some classes). The algorithm identifies both types of features and is shown to perform better than Zhu et al. (2007) on synthetic and microarray gene expression data.

4. Stage 2: Modelling GRNs from time series data

An overview of existing EAs for GRN model inference from time series data is presented in this section. The discussion considers methods applied to both discrete and continuous models. Due to added complexity of the latter models, many EA approaches have been developed, from classical to advanced algorithms, and these are outlined, indicating their gradual development and their role in GRN inference.

4.1 Discrete models

Although applied more extensively for continuous models, evolutionary algorithms have been used for qualitative model analysis also. Linden & Bhaya (2007) introduced a method of inferring fuzzy rules from microarray data, using genetic programming. The algorithm uses the reverse Polish notation[6] for rules, which can be easily represented as trees, with three Boolean operators for the conditions: NOT, AND and OR. A population of this type was evolved using classical genetic operators on trees and the best individuals were selected to progress to the next generation. Fitness was defined as the percentage error observed between real data and the data generated by the rules. The algorithm was applied to finding rules in microarray data from experiments on the response to cold of the plant *Arabidopsis Thaliana*, as well as on the rat nervous system. A clustering algorithm was applied initially to reduce dimensionality, with resulting clusters considered to form one node in the network. Results were validated based on previous knowledge of the datasets, while new hypotheses for subsequent laboratory experimentation were proposed.

In Repsilber et al. (2002) a genetic algorithm was used to fit a multistate discrete network, (Section 2.2.1), to simulated gene expression data. The aim was to rank previously known hypotheses about the structure of the network, by allowing model parameters to evolve. The approach also introduced time delays ($\delta = \{\delta_1, .., \delta_N\}$) to model the *time gap* between the transcription of one gene and the regulation effect of the resulting protein. This time gap is the time needed for the mRNA to be translated into a protein, so a node changes its state only after initiation of transcription *plus a time delay*. The algorithm thus searches for the most probable model structure for the data available.

Another method of inferring discrete GRN models, based on genetic programming, was developed by Eriksson & Olsson (2004). Here, genes take Boolean values and the regulatory network structure for each target gene is encoded as a tree and evolved to obtain better structure. For each such tree in the population, a Boolean function is determined from data, (by computing the truth table using expression levels in the data), and fitness is assigned based on ambiguities that arise. This results in choosing those structures that have fewer ambiguities, so indicate more plausible interactions. The method was tested on synthetic networks of different size, (10 to 160 genes), and shown, for networks smaller than 40 genes to successfully locate structures with over 75% of the optimal fitness, (with a value of 51% for the 160-gene network). However, to date, this method has not been validated with real data.

A similar evolutionary algorithm optimising the *wiring of a Boolean network* is described in Esmaeili & Jacob (2009). This method starts with randomly generated wirings with a limited number of regulators for each gene and evolves these structures using differential evolution.

[6] In the reverse Polish notation, the symbol order in an expression is changed: the operators are in front of the operands. For instance, $a \times (b + c) - d$ becomes $- \times a + bcd$.

Method	EA	Mo-del	Fitness	Local search	Sta-ges	Datasets (Size)
Ando et al. (2002); Iba (2008); Sakamoto & Iba (2001)	GP	ODE	Error + degree penalty	LMS	-	Synthetic (5)
Fomekong-Nanfack et al. (2007)	ES	PDE	Error	-	-	Fly (6)
Ando & Iba (2003)	GA	LDE	Error	-	$\surd$	E. coli (9), Yeast (8)
Deng et al. (2005)	GA	LDE	1 + Error	-	$\surd$	Rat 20)
Tominaga et al. (1999)	GA	SS	Error	-	-	Synthetic (2)
Iba & Mimura (2002)	GA	SS	Error	-	$\surd$	Synthetic (10)
Kikuchi et al. (2003)	GA	SS	Error + parameter penalty	Simplex Crossover	$\surd$	Synthetic (5)
Kimura et al. (2003)	GA	SS	Error + parameter penalty	QP	$\surd$	Synthetic (30)
Noman & Iba (2005)	DE	SS	Error + parameter penalty	-	$\surd$	Synthetic (5)
Noman & Iba (2006; 2007)	DE	SS	Error + parameter penalty	HC	$\surd$	Synthetic (20), Yeast (14-qualitative)

Table 2. Evolutionary algorithms for continuous model inference (1). *Error* is a measure of the difference between observed and simulated data, and different versions of this (RSS, MSE) have been used; however, their use is equivalent, as the number of genes and time points, (i.e. degrees of freedom), is the same for all individuals to be evaluated in a given optimisation run. Methods employing any type of iterated optimisation (Section 4.2.2), nested optimisation (Section 4.2.1) or *divide et impera* (Section 4.2.1) contain $\surd$ in column *Stages*. Abbreviations: ODE - ordinary differential equations, EA - evolutionary algorithm, GP - genetic programming, LMS - least means squares, PDE - partial differential equations, LDE - linear differential equations, SS - S-System, ES - evolutionary strategy, DE - differential evolution , HC - hill climbing, QP- quadratic programming.

Each structure is evaluated using a multi-objective approach, which aims at optimising sensitivity, attractor cycle length and number of attractors. The method is shown to yield more stable structures for a synthetic network of size 8. However as before, the method was not applied to real gene expression data, so further analysis is required.

4.2 Continuous models

Several algorithms for inference of continuous GRN models from gene expression data have been developed in recent years, and Tables 2 and 3 give an overview of methods. These include application of classical evolutionary techniques and development of novel algorithms, especially tailored for gene expression data.

One of the first approaches to GRN reverse engineering, based on evolutionary computation, is that of Tominaga et al. (1999). A classic, double-encoded genetic algorithm is used to

Method	EA	Mo-del	Fitness	Local search	Sta-ges	Datasets (Size)
Xu et al. (2007)	DE PSO	RNN	Error	-	-	Synthetic (8), E. Coli(8)
Koduru et al. (2007; 2004; 2005; 2008)	GA PSO	LDE, SS, RNN	Multi Objective - Error per gene	Simplex	-	Rice (2), A. Thaliana(3)
Imade et al. (2004; 2003); Morishita et al. (2003); Ono et al. (2004)	GA	SS	Error	GA	√	Synthetic (5)
Spieth, Streichert, Supper, Speer & Zell (2005); Spieth et al. (2004)	GA	SS	Error	ES	√	Synthetic (20)
Keedwell & Narayanan (2005)	GA	ANN	BP Error	BP	√	Synthetic Boolean (10), Rat (112), Yeast (2468)
Daisuke & Horton (2006)	GA	SS	Error	Simplex Crossover, Scale free	√	Synthetic (5), Mouse (7)
Spieth, Streichert, Speer & Zell (2005b)	GA, ES	SS	Multi Objective - Error, Connectivity	-	-	Synthetic (5, 10)
Kabir et al. (2010)	SA-DE	LTV	Error	-	-	Synthetic (5), E. Coli (6)

Table 3. Evolutionary algorithms for continuous model inference (2). Abbreviations: LDE - linear differential equations, SS - S-System, PSO - particle swarm optimisation, RNN - recurrent neural network, ES - evolutionary strategy, ANN - artificial neural network, BP - back-propagation, SA-DE - self adaptive differential evolution, LTV - linear time-variant.

infer S-System models from time series data. However, this method was only applied to synthetic data for a very small network (2 genes). Another more recent application of a classic evolutionary algorithm is that of Fomekong-Nanfack et al. (2007). This employs an evolutionary strategy to optimise model parameters for a 6-gene developmental network for *Drosophila Melanogaster*, based on partial differential equations, (reaction-diffusion model, Section 2.2.1). Although suitable for a larger network than Tominaga et al. (1999), the size of the inferred GRN is still very small compared to the total number of genes typically involved in such a system, showing that classical evolutionary algorithms are not powerful enough for larger networks. This was also indicated in Sîrbu et al. (2010a), where an empirical comparison was performed between different evolutionary algorithms, including Tominaga et al. (1999), and which showed that only hybrid evolutionary algorithms scaled to larger systems.
The limitation in size and model quality for quantitative analysis derives from the nature of the data to be studied. Typically, gene expression time series are too short, due to experimental limitations, while the number of parameters needed to describe quantitative models is very large. This creates an *under-determination* problem, so that multiple models can have very similar simulated behaviour, with corresponding poor reliability for inferential algorithms. Additionally these data are intrinsically noisy, hence application of algorithms to real data is not straightforward, (Sîrbu et al., 2010a). This is emphasised in Tables 2 and

3, where many algorithms are seen not to have been applied to real data. A consequence of noise and under-determination is the *ruggedness* of the fitness landscape[7] for this problem, (Rodrigo et al., 2010). However, evolutionary algorithms are known to perform well on underdetermined problems and noisy fitness functions (Mitchell, 1999), so have clear benefit over other inferential methods. Evolutionary computation approaches that address these issues can guide the optimisation towards more plausible solutions and are discussed next.

4.2.1 Addressing the under-determination problem

Divide et impera

Given that model parameters are independent for each gene, (relying only on the expression level of the genes at previous time points), one method of addressing under-determination is to use a *divide et impera* approach. This consists of separate optimisation of parameters for each individual gene, using observed rather than simulated expression levels for the other genes. This method has been implemented in several evolutionary algorithms, (Ando & Iba, 2003; Iba & Mimura, 2002; Keedwell & Narayanan, 2005; Liu et al., 2008; Noman & Iba, 2006; 2007), and has the advantage of reducing the solution space by decreasing the number of parameters to be inferred at any one time. In a recent comparison study, Sîrbu et al. (2010a), algorithms using this approach scaled better than those which sought to optimise parameters for the entire network simultaneously. However, a disadvantage of this method is that, typically, expression levels in the data are noisy, and these are used in single gene simulations, resulting in model parameters slightly different from those that would be obtained by simulating all genes. In consequence, models obtained by combining single gene solutions may not fit the data very well. This can be avoided by a second optimisation stage: starting from single gene models, evolutionary optimisation is employed to fine-tune the parameters for the entire network, (e.g. Ando & Iba (2003); Noman & Iba (2005)). Further, a model that handles noise better, such as an artificial neural network, (which employs a sigmoid function to compute expression levels), may also decrease this effect.

Obtaining skeletal/ scale free structures

To further distinguish between many possible models, known characteristics of the network structure, such as low connectivity or scale-free nature, have been considered also. Many methods apply such knowledge to the optimisation process at different levels of the algorithm. The simplest idea sets parameters to zero once these fall below a fixed threshold (Kikuchi et al., 2003; Tominaga et al., 1999). However, more advanced approaches have also been developed. For instance, Kikuchi et al. (2003); Kimura et al. (2003); Noman & Iba (2005; 2006) use an additional term that penalises solutions with large parameter values. A refinement of this penalty-based idea can be seen in methods, which start by penalising all connections, and then use a connectivity threshold to reduce possibilities. This results in more advanced fitness functions, and is possible because evolutionary optimisation, unlike numerical methods, has the advantage of not restricting fitness function type. A similar method, (Spieth, Streichert, Speer & Zell, 2005b), uses the connectivity as a second objective in multi-objective optimisation. Analogously, Ando et al. (2002); Iba (2008); Sakamoto & Iba (2001) have used genetic programming to evolve sparse ordinary differential equations, by penalising functions of large degree. Deng et al. (2005) also employed a limit on the connectivity of a linear differential equation model, evolving the connectivity parameter during optimisation, rather

[7] In evolutionary algorithms, each candidate solution can be considered a point in a multidimensional space (i.e. the solution space), with a corresponding fitness value. The fitness values for all points in the solution space generate the so called fitness landscape.

than fixing it initially, in order to find the optimal connectivity for the network, i.e. the number of regulators that achieves best data fit.
Another mechanism, used to obtain solutions with more plausible structures, is local search. For instance, Noman & Iba (2006) use hill climbing to set as many parameters to zero in two candidate solutions for each generation. Also, in Daisuke & Horton (2006), models are checked for scale-free structure, and modified if they do not comply, by adding or removing random connections, (setting the corresponding parameters to 0). All these methods result in sparser networks, as unnecessary parameters are set to zero.

Nested optimisation

Reduction in the number of parameters to be optimised has been also performed using a nested optimisation approach, (Keedwell & Narayanan, 2005; Morishita et al., 2003; Spieth, Streichert, Supper, Speer & Zell, 2005; Spieth et al., 2004). These methods divide the search into two stages: structure and parameter search. During structure search, network topology is evolved using a genetic algorithm. Candidate structures are built so that the number of regulators is bounded for each gene, and these are evaluated by a second algorithmic stage, which optimises parameters for the existing connections. This reduces the number of real-valued parameters to be inferred at the second stage. The parameter search is performed using an evolution strategy, (Spieth, Streichert, Supper, Speer & Zell, 2005; Spieth et al., 2004), a genetic algorithm, (Morishita et al., 2003), or back-propagation, (Keedwell & Narayanan (2005), with an artificial neural network as the model). Again, this is facilitated by the flexibility of fitness evaluation, which is characteristic of evolutionary algorithms. Nested optimisation increases parallelisation potential, (a parallelised version of Morishita et al. (2003) was later developed by Imade et al. (2003)). Separation of structure and parameter search is important as this allows the topology to have a larger influence in the optimisation process, rather than optimising real-valued parameters directly. This is particularly relevant in the current context as dynamical behaviour in biological networks relies mostly on topology, (Alvarez-Buylla et al., 2007), with parameter perturbations of lesser importance.

Parallelisation

Quantitative models require optimisation of a very large number of parameters, and fitness evaluation is costly in simulation terms. These costs increase when additional time series datasets are used, so parallelisation of methods is mandatory. Evolutionary algorithms have the advantage of being intrinsically parallel, facilitating efficient multi-threading of the optimisation process. Several examples of parallel implementations exist in evolutionary methods for GRN modelling, (Daisuke & Horton, 2006; Fomekong-Nanfack et al., 2007; Imade et al., 2004; 2003; Spieth, Streichert, Speer & Zell, 2005a). These correspond to both grid and cluster systems, while parallel frameworks for analysis have been implemented and are publicly available (Spieth et al., 2006; Swain et al., 2005).

4.2.2 Handling local minima

Combining multiple methods

Due to the ruggedness of the fitness landscape, an evolutionary algorithm can be trapped in local minima, and fail to find an optimal solution. One approach that seeks to avoid this is to combine different evolutionary methods. This is facilitated by their simplicity and flexibility. For instance, Xu et al. (2007) alternates differential evolution and *particle swarm optimisation*, (parameterisation of a neural network model), to obtain better overall models than from the two optimisation strategies separately. A different approach, (Daisuke & Horton, 2006; Kikuchi et al., 2003), uses *Simplex crossover*, which efficiently balances the exploration and exploitation of the search space (Kikuchi et al., 2003), and in consequence speeds up convergence, (as the local minima problem is diminished).

Iterated optimisation

A second technique with the same aim is iterated optimisation, (possible due to the stochastic nature of evolutionary computation). Multiple runs of the same algorithm typically lead to different solutions, i.e. different local minima, which can be combined to obtain a better model: Kikuchi et al. (2003) describe a second optimisation run, initialised with these local solutions. An alternative is to analyse local solutions by methods other than evolutionary algorithms. For instance, Daisuke & Horton (2006); Deng et al. (2005) employ a *voting procedure* for connections found in multiple runs, while Noman & Iba (2005) use voting to find null parameters in the model. Similarly, Noman & Iba (2007) apply Z-score analysis to local solutions to find plausible qualitative connections for yeast cell cycle data (quantitative analysis being hampered by data limitations of length and noise).

4.2.3 Handling noise

Noise is a serious problem in gene expression measurements and, unfortunately, most of the algorithms developed for model inference from these data do not specifically take it into account. This makes many methods unfit for real data, even when validated in principle for synthetic systems. A recent comparison of evolutionary methods for quantitative model inference has enabled evaluation of method performance on noisy data, (Sîrbu et al., 2010a). While most methods give good behaviour up to 5% added noise, only two maintained this with up to 10%. One, (Keedwell & Narayanan, 2005), uses an artificial neural network to model gene regulation, while the second employs a local search procedure, based on Quadratic Programming, that handles noisy measurements, (Kimura et al., 2003). The superior performance of these two methods is a strong indication that noise needs to be explicitly addressed in the model or evolutionary process, in order to obtain algorithms that can be applied to real-world data. (Refer again to Tables 2 and 3)

5. Stage 3: Data integration for GRN modelling

Evolutionary approaches for data integration are few, so far, even though the need for inclusion of other types of data in the inferential process, (due to the under-determination problem), has been widely acknowledged, (Hecker et al. (2009) and references therein). One of the first methods attempting to incorporate previous knowledge is that of Shin & Iba (2003), where the AIC-based[8] fitness function, (similar to that of Noman & Iba (2006)), is modified to account for known interactions between genes in an S-System model. Thus models containing known interactions have better fitness and this leads the search towards regions in space that are more likely to contain the correct structure. The Shin & Iba (2003) algorithm was applied repeatedly and results were analysed using Z-scores to identify significant relationships. On synthetic data, the approach was shown to have increased sensitivity in finding correct interactions, compared to the standard method, which made no use of previous knowledge. Further, the former worked well even when previous knowledge was partially incorrect, (not unusual in a real experiment). When applied to the real microarray data of *E. Coli*, the method was also shown to identify previously known interactions.
A second data type integrated into the evolutionary optimisation process relates to knock-out experiments. Ono et al. (2004) attempted inclusion of time series knock-out data, and

[8] Akaike's Information Criterion (AIC, Noman & Iba (2006)) is an information criterion used for model selection, which is based on the error between observed and simulated data.

demonstrated that this improved the structure search. Again, the method was only applied to synthetic systems. However, Ferrazzi et al. (2007) integrated steady state knock-out measurements to infer parameters for a linear model of regulation in the cell cycle of *Saccharomyces Cerevisiae*. The additional data are used to initialise a genetic algorithm with biologically plausible interactions, by analysing differentially expressed genes in the knock-out experiments, and keeping these known interactions fixed in the structure. This enhanced approach was compared to a simple genetic algorithm, and was shown to be more robust. Thus, feeding the optimisation with interactions from knock-out data guided the algorithm towards similar solutions in the search space during different runs, (implying that these were closer to the real network of interactions).

5.1 Large scale data integration

To date, only one additional piece of knowledge data has been added to the evolutionary optimisation process. However, combining information sources could bring significant improvement. Consequently, we are developing a novel integrative evolutionary algorithm that aims at combining multiple data sources for GRN modelling. The algorithm uses an ANN as a model and is based on the idea introduced by Keedwell & Narayanan (2005) of dividing the structure and parameter optimisation process. A genetic algorithm is used for structure search, and back-propagation for parameter search and structure evaluation. As this is a *divide et impera* approach, a second optimisation stage has been added to the original algorithm, which performs fine tuning of models, obtained by combining single gene results. This second stage is executed after iterating the first stage several times for each gene with different connectivity limits (i.e. different number of regulators), so each gene connectivity is optimised to obtain structures that are better able to simulate the data. Additionally, the algorithm allows for reducing the list of possible regulators to a user defined list, (from previously known transcription factors, if available). This further decreases the search space and allows introduction of meta-information into the algorithm.

A first step for heterogeneous data integration is combining gene expression data from different sources. Although time series from one laboratory are typically short, multiple data sets from different sources, which nevertheless measure the same process, are available. A study of cross-platform microarray data integration has been performed, (Sîrbu et al., 2010c), and has demonstrated that models obtained from combined data are both more robust to noise and parameter perturbation, and display less noise over-fitting. Additionally, normalisation methods for multiple microarray platforms have been analysed in the context of GRN modelling (Sîrbu et al., 2010b).

It appears that time series data integration does reduce the under-determination problem, but it makes the inferential process much more computationally intensive, (especially for evolutionary optimisation, as multiple series have to be simulated for each fitness evaluation). To address this, a fine-grained parallelisation of the algorithm has been performed. This uses a different processor to evaluate each candidate solution in the population. Additionally, during the first stage, individual gene runs are also divided between multiple processors.

A second step for data integration, as seen earlier, is that of using steady state knock-out experiments. These are used here not only for initialisation, as in Ferrazzi et al. (2007), but also for model evaluation. Unlike Ferrazzi et al. (2007), the structure extracted from these data is not fixed, so the algorithm can change this during optimisation, in order to select the interactions that give better data fit. For evaluation, the models are simulated for a user-defined number of time points, to reach the steady state, using null expression values for the knocked out genes and starting with expression values of 0.5, (middle of interval $(0, 1)$, which is the range of gene expression values), for the rest. The error between the resulting

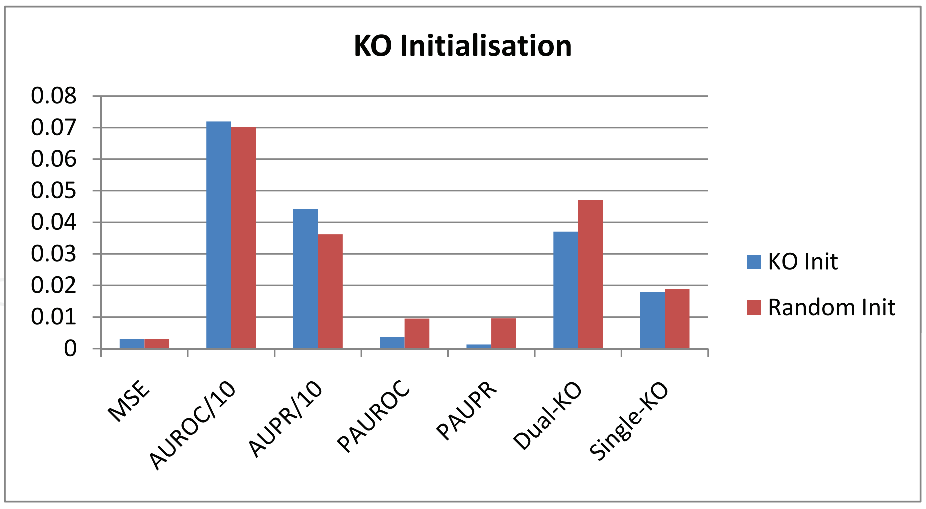


Fig. 3. Initialisation with knock-out (KO) experiments. Graph shows average values over three runs for each experiment: KO Init (initialisation with knock-out experiments) and Random Init (random initialisation). Qualitative results are displayed using AUROC (area under ROC curve), AUPR (area under precision-recall curve), PAUROC (p-value of AUROC) and PAUPR (p-value of AUPR). Quantitative results show MSE values between data and simulation for knock-out (Dual KO and Single KO) and wild-type (MSE) experiments. Results obtained using KO Init are better both quantitatively and qualitatively.

state and the data is used as an additional term in refining the fitness function. This evaluation is only performed during the second stage of the algorithm, when the model for the entire network is optimised, as it requires simulated values for all the genes in the network.
We have validated this approach with simulated gene expression data from the DREAM 4 challenge, (Marbach et al., 2009). Although these are synthetic data, they are generated using models inspired by real GRNs,(as provided by the authors), so have plausible structures. Also, the data contain added noise, in an effort to mimic real gene expression data. We have compared models, which use both knock-out initialisation and fitness evaluation, to those from the simple version of the algorithm, and results are shown in Figures 3 (for initialisation) and 4 (for evaluation). These show that adding initialisation with knock-out experiments leads to models containing more correct interactions, (higher AUPR, AUROC and lower p-values), which can simulate known dynamical behaviour, (giving lower MSE values in both wild type and knock-out simulations). If knock-out experiments are used for evaluation, simulation of wild-type experiments is disimproved (higher MSE), but this may be due to residual noise, given the improvement in dual and single knock-out simulations and in qualitative results, (more correct interactions). Considering the promising results for synthetic systems, validation of the method against real data is clearly required.
Given that structure is important in simulating GRN behaviour, with currently used fitness functions, (based on error between observed and simulated data), unable to reward those models that can reproduce these, albeit with shifts in expression values, an additional evaluation term has been introduced, (Sîrbu et al., 2010d). This measures the Pearson correlation between observation and simulation. Additionally, inclusion of this term in back-propagation, allows us to find parameters that minimise the error and maximise the correlation. In comparison to the initial algorithm, (incorporating error-based fitness and back-propagation), the modified approach is found to yield better structures (with higher

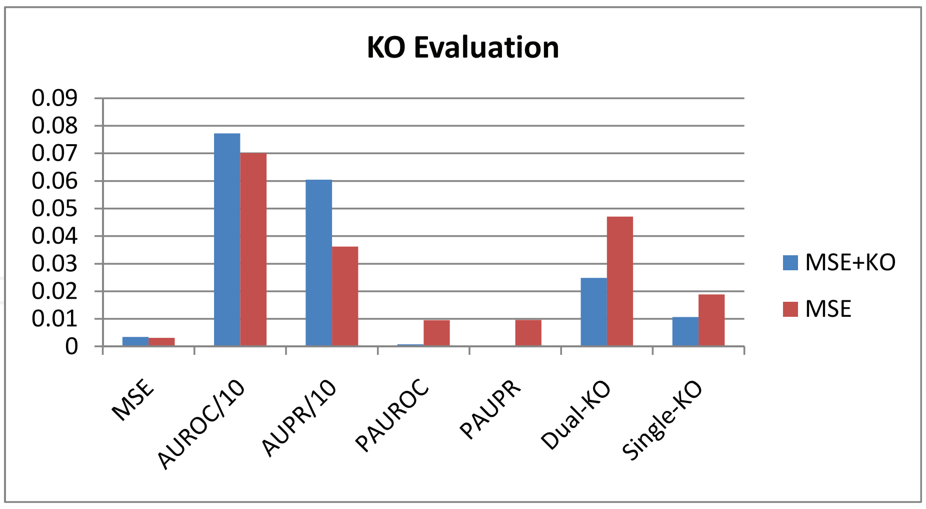


Fig. 4. Evaluation with knock-out experiments. Average performance over three runs for each experiment: MSE+KO (with knock-out evaluation) and MSE (without knock-out evaluation). Qualitative and quantitative results are analysed using the same criteria as in Figure 3. These indicate better behaviour for models obtained from MSE+KO.

AUPR and AUROC) for synthetic data (from DREAM4) and more plausible interactions in real microarray data (Yeast cell cycle). Further details are given in Sîrbu et al. (2010d).
Extension of the integrative framework is anticipated for future work, which should involve inclusion of promoter sequence and binding affinity data (ChIP-chip), as well as incorporation of RNA-seq and RNA-interference measurements in model evaluation and inference process.

6. Conclusions

This chapter has presented the role of evolutionary algorithms at different stages of gene regulatory network inference. These include (i) expression pattern analysis, (ii) model inference from time series data and (iii) data integration for model inference. For (i), methods for clustering and feature selection for gene expression data have been described. For (ii), method development from classical to more advanced hybrid algorithms has been presented. This has been motivated by issues in network modelling, such as under-determination and noisy data. These issues have been addressed to some extent by taking advantage of the flexibility and power of evolutionary approaches. For instance, the flexibility of the fitness function has been used to reward models with sparse or scale free structures. Hybridisation with local search and other optimisation algorithms has also benefited from the simple basis of the evolutionary algorithmic scheme, in order to avoid local minima traps and to handle noise. Additionally, the parallelisation potential of these methods, combined with their stochastic underpinning, has led to iterated algorithm versions, (designed to handle local solutions), and nested optimisation, (used to limit the number of real-valued parameters to be addressed). All these improvements have permitted a scale-up of quantitative modelling, from 2 to 30 genes. However, this is still very modest compared to real GRN requirements.
Many of the methods presented have, to date, been applied only to synthetic data, while most applications to real data can yield only qualitative results, as quantitative models obtained remain unreliable. In order to further improve inference, different data sources can be combined, and this has been presented as the third stage of GRN inference. Advances in high-throughput technologies other than microarrays and global research efforts have created

very large biological data sets containing protein-protein interactions, protein measurements, knock-out experiments, protein binding sites and gene sequence information. Although current such data are insufficient to determine the underlying GRN, combining them could prove to be very powerful and EAs are flexible enough to enable their integration. Existing methods, nevertheless, under-exploit EC potential, to some extent, by integrating only one additional data type. In consequence, we have outlined here the basis for a novel framework, which is being developed and which aims at large-scale data integration for GRN quantitative modelling. This uses fitness evaluation, initialisation and parallelisation to include heterogeneous data and knock-out experiments in the optimisation process. The framework will be extended in the future to employ promoter sequences and protein binding affinities, (such as those extracted from ChIP-Chip data), for model evaluation, and will integrate RNA-seq data and RNA-interference measures in the inferential process.

7. Acknowledgements

We wish to acknowledge the SFI/HEA Irish Centre for High-End Computing (ICHEC) for the provision of computational facilities and assistance. This work has been conducted with support from the Irish Research Council for Science Engineering and Technology EMBARK Initiative.

8. References

Akutsu, T., Miyano, S. & Kuhara, S. (2000). Inferring qualitative relations in genetic networks and metabolic pathways, *Bioinformatics* **16**(8): 727–734.
URL: *http://bioinformatics.oxfordjournals.org/cgi/content/abstract/16/8/727*

Alvarez-Buylla, E. R., Benitez, M., Davila, E. B., Chaos, A., Espinosa-Soto, C. & Padilla-Longoria, P. (2007). Gene regulatory network models for plant development, *Current Opinion in Plant Biology* 10(1): 83 – 91.

Ando, S. & Iba, H. (2003). Estimation of gene regulatory network by genetic algorithm and pairwise correlation analysis, *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on* 1: 207–214.

Ando, S., Sakamoto, E. & Iba, H. (2002). Evolutionary modeling and inference of gene network, *Inf. Sci. Inf. Comput. Sci.* 145(3-4): 237–259.

Baeck, T., Fogel, D. B. & Michalewicz, Z. (2000). *Evolutionary Computation 1: Basic Algorithms and Operators*, Institute of Physics Publishing Bristol and Philadelphia.

Baldi, P. & Hatfield, W. (2002). *DNA Microarray and Gene Expression. From experiments to data analysis and modeling*, Cambridge University Press.

Bar-Joseph, Z. (2004). Analyzing time series gene expression data, *Bioinformatics* 20(16): 2493–2503.

Brown, T. (2002). *Genomes*, BIOS Scientific publishers Ltd.

Bryan, K. (2005). Biclustering of expression data using simulated annealing, *CBMS '05: Proceedings of the 18th IEEE Symposium on Computer-Based Medical Systems*, IEEE Computer Society, Washington, DC, USA, pp. 383–388.

Chakraborty, A. & Maka, H. (2005). Biclustering of gene expression data using genetic algorithm, *Computational Intelligence in Bioinformatics and Computational Biology, 2005. CIBCB '05. Proceedings of the 2005 IEEE Symposium on*, pp. 1 – 8.

Cheng, Y. & Church, G. M. (2000). Biclustering of expression data, *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, AAAI Press, pp. 93–103.

Daisuke, T. & Horton, P. (2006). Inference of scale-free networks from gene expression time series, *Journal of Bioinformatics and Computational Biology* 4: 503–514.

Deng, X., Geng, H. & Ali, H. (2005). Examine: a computational approach to reconstructing gene regulatory networks, *Biosystems* 81: 125–136.

Di Gesu, V., Giancarlo, R., Lo Bosco, G., Raimondi, A. & Scaturro, D. (2005). Genclust: A genetic algorithm for clustering gene expression data, *BMC Bioinformatics* 6(1): 289. URL: *http://www.biomedcentral.com/1471-2105/6/289*

Eriksson, R. & Olsson, B. (2004). Adapting genetic regulatory models by genetic programming, *Biosystems* 76(1-3): 217 – 227. Papers presented at the Fifth International Workshop on Information Processing in Cells and Tissues.

Esmaeili, A. & Jacob, C. (2009). A multi-objective differential evolutionary approach toward more stable gene regulatory networks, *Biosystems* 98(3): 127 – 136.

Ferrazzi, F., Magni, P., Sacchi, L., Nuzzo, A., Petrovic, U. & Bellazzi, R. (2007). Inferring gene regulatory networks by integrating static and dynamic data, *International Journal of Medical Informatics* 76(Supplement 3): S462 – S475.

Fomekong-Nanfack, Y., Kaandorp, J. A. & Blom, J. (2007). Efficient parameter estimation for spatio-temporal models of pattern formation: case study of Drosophila melanogaster, *Bioinformatics* 23(24): 3356–3363. URL: *http://bioinformatics.oxfordjournals.org/cgi/content/abstract/23/24/3356*

He, F., Balling, R. & Zeng, A.-P. (2009). Reverse engineering and verification of gene networks: Principles, assumptions, and limitations of present methods and future perspectives, *Journal of Biotechnology* 144(3): 190 – 203. Systems Biology for Biotechnological Innovation.

Hecker, M., Lambeck, S., Toepfer, S., van Someren, E. & Guthke, R. (2009). Gene regulatory network inference: Data integration in dynamic models–a review, *Biosystems* 96(1): 86 – 103.

Hurd, P. J. & Nelson, C. J. (2009). Advantages of next-generation sequencing versus the microarray in epigenetic research, *Briefings in Functional Genomics & Proteomics* 8(3): 174–183. URL: *http://bfg.oxfordjournals.org/content/8/3/174.abstract*

Iba, H. (2008). Inference of differential equation models by genetic programming, *Information Sciences* 178(23): 4453–4468.

Iba, H. & Mimura, A. (2002). Inference of a gene regulatory network by means of interactive evolutionary computing, *Information Sciences* 145(3-4): 225–236.

Imade, H., Mizuguchi, N., Ono, I., Ono, N. & Okamoto, M. (2004). "gridifying" an evolutionary algorithm for inference of genetic networks using the improved goga framework and its performance evaluation on obi grid, *LSGRID*, pp. 171–186.

Imade, H., Morishita, R., Ono, I., Ono, N. & Okamoto, M. (2003). A grid-oriented genetic algorithm for estimating genetic networks by s-systems, *SICE 2003 Annual Conference* 3: 2750–2755 Vol.3.

Kabir, M., Noman, N. & Iba, H. (2010). Reverse engineering gene regulatory network from microarray data using linear time-variant model, *BMC Bioinformatics* 11(Suppl 1): S56. URL: *http://www.biomedcentral.com/1471-2105/11/S1/S56*

Keedwell, E. & Narayanan, A. (2005). Discovering gene networks with a neural-genetic hybrid, *Computational Biology and Bioinformatics, IEEE/ACM Transactions on* 2(3): 231–242.

Kerr, G., Ruskin, H. J., Crane, M. & Doolan, P. (2008). Techniques for clustering gene expression data, *Computers in Biology and Medicine* 38(3): 283–293.

Kikuchi, S., Tominaga, D., Arita, M., Takahashi, K. & Tomita, M. (2003). Dynamic modeling of genetic networks using genetic algorithm and S-system, *Bioinformatics* 19(5): 643–650. URL: *http://bioinformatics.oxfordjournals.org/cgi/content/abstract/19/5/643*

Kimura, S., Hatakeyama, M. & Konagaya, A. (2003). Inference of S-system models of genetic networks using a genetic local search, *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on* 1: 631–638 Vol.1.

Koduru, P., Das, S. & Welch, S. M. (2007). Multi-objective hybrid pso using μ-fuzzy dominance, *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, ACM, New York, NY, USA, pp. 853–860.

Koduru, P., Das, S., Welch, S. & Roe, J. L. (2004). Fuzzy dominance based multi-objective GA-Simplex hybrid algorithms applied to gene network models, *Genetic and Evolutionary Computation - GECCO 2004*, pp. 356–367.

Koduru, P., Das, S., Welch, S., Roe, J. L. & Lopez-Dee, Z. P. (2005). A co-evolutionary hybrid algorithm for multi-objective optimization of gene regulatory network models, *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, ACM, New York, NY, USA, pp. 393–399.

Koduru, P., Dong, Z., Das, S., Welch, S., Roe, J. L. & Charbit, E. (2008). A multiobjective evolutionary-simplex hybrid approach for the optimization of differential equation models of gene networks, *IEEE Trans. Evolutionary Computation* 12(5): 572–590.

Lee, W.-P. & Tzou, W.-S. (2009). Computational methods for discovering gene networks from expression data, *Briefings in Bioinformatics* 10(4): 408–423. URL: *http://bib.oxfordjournals.org/cgi/content/abstract/10/4/408*

Lee, W.-P. & Yang, K.-C. (2008). A clustering-based approach for inferring recurrent neural networks as gene regulatory networks, *Neurocomputation* 71(4-6): 600–610.

Li, L. (2001). Gene assessment and sample classification for gene expression data using a genetic algorithm/k-nearest neighbor method, *Combinatorial chemistry and high throughput screening* 4: 727.

Li, L., Umbach, D. M., Terry, P. & Taylor, J. A. (2004). Application of the ga/knn method to seldi proteomics data, *Bioinformatics* 20(10): 1638–1640.

Liang, S., Fuhrman, S. & Somogyi, R. (1998). Reveal: a general reverse engineering algorithm for inference of genetic network architectures, *Pacific Symposium on Biocomputing* pp. 18–29.

Linden, R. & Bhaya, A. (2007). Evolving fuzzy rules to model gene expression, *Biosystems* 88(1-2): 76 – 91.

Liu, H., Li, J. & Wong, L. (2002). A Comparative Study on Feature Selection and Classification Methods Using Gene Expression Profiles and Proteomic Patterns, *Genome Informatics* 13.

Liu, K.-H., Zhang, J., Li, B. & Du, J.-X. (2009). A ga-based approach to ica feature selection: An efficient method to classify microarray datasets, *ISNN 2009: Proceedings of the 6th International Symposium on Neural Networks*, Springer-Verlag, Berlin, Heidelberg, pp. 432–441.

Liu, X., Fu, J., Gu, D., Liu, W., Liu, T., Peng, Y., Wang, J. & Wang, G. (2008). Genome-wide analysis of gene expression profiles during the kernel development of maize (zea mays l.), *Genomics* 91(4): 378 – 387.

Logan, J., Edwards, K. & Saunders, N. (eds) (2007). *Real-Time PCR: Current Technology and Applications*, Caister Academic Press.

Lu, Y., Lu, S., Fotouhi, F., Deng, Y. & Brown, S. (2004a). Fgka: A fast genetic k-means algorithm: March 2004., *Proceedings of the 2004 ACM Symposium on Applied Computing (SAC)*.

Lu, Y., Lu, S., Fotouhi, F., Deng, Y. & Brown, S. (2004b). Incremental genetic k-means algorithm and its application in gene expression data analysis, *BMC Bioinformatics* 5(1): 172. URL: *http://www.biomedcentral.com/1471-2105/5/172*

Maki, Y., Tominaga, D., Okamoto, M., Watanabe, S. & Eguchi, Y. (2001). Development of a system for the inference of large scale genetic networks., *Pacific Symposium on Biocomputing* pp. 446–458. URL: *http://view.ncbi.nlm.nih.gov/pubmed/11262963*

Manning, C. D. & Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*, MIT Press, chapter 14.

Marbach, D., Schaffter, T., Mattiussi, C. & Floreano, D. (2009). Generating realistic in silico gene networks for performance assessment of reverse engineering methods., *Journal of Computational Biology* 16(2): 229–239. URL: *http://dx.doi.org/10.1089/cmb.2008.09TT*

Mitchell, M. (1999). *An Introduction to Genetic Algorithms*, The MIT Press.

Mitra, S. & Banka, H. (2006). Multi-objective evolutionary biclustering of gene expression data, *Pattern Recognition* 39(12): 2464 – 2477.

Morishita, R., Imade, H., Ono, I., Ono, N. & Okamoto, M. (2003). Finding multiple solutions based on an evolutionary algorithm for inference of genetic networks by S-system, *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on* 1: 615–622 Vol.1.

Noman, N. & Iba, H. (2005). Inference of gene regulatory networks using s-system and differential evolution, *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, ACM, New York, NY, USA, pp. 439–446.

Noman, N. & Iba, H. (2006). Inference of genetic networks using S-system: information criteria for model selection, *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, ACM, New York, NY, USA, pp. 263–270.

Noman, N. & Iba, H. (2007). Inferring gene regulatory networks using differential evolution with local search heuristics, *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 4(4): 634–647.

Ono, I., Seike, Y., Morishita, R., Ono, N., Nakatsui, M. & Okamoto, M. (2004). An evolutionary algorithm taking account of mutual interactions among substances for inference of genetic networks, *Evolutionary Computation, 2004. CEC2004. Congress on* 2: 2060–2067 Vol.2.

Ooi, C. H. & Tan, P. (2003). Genetic algorithms applied to multi-class prediction for the analysis of gene expression data, *Bioinformatics* 19(1): 37–44.

Pal, S., Bandyopadhyay, S. & Ray, S. (2006). Evolutionary computation in bioinformatics: a review, *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 36(5): 601–615.

Przytycka, T. M., Singh, M. & Slonim, D. K. (2010). Toward the dynamic interactome: it's about time, *Briefings in Bioinformatics* 11(1): 15–29. URL: *http://bib.oxfordjournals.org/cgi/content/abstract/11/1/15*

Repsilber, D., Liljenström, H. & Andersson, S. G. E. (2002). Reverse engineering of regulatory networks: simulation studies on a genetic algorithm approach for ranking hypotheses, *Biosystems* 66(1-2): 31 – 41.

Robnik-Łikonja, M. & Kononenko, I. (2003). Theoretical and empirical analysis of relieff and rrelieff, *Machine Learning* 53: 23– 69.

Rodrigo, G., Carrera, J. & Elena, S. F. (2010). Network design meets in silico evolutionary biology, *Biochimie* 92(7): 746 – 752.

Sakamoto, E. & Iba, H. (2001). Inferring a system of differential equations for a gene regulatory network by using genetic programming, *Proceedings of Congress on Evolutionary Computation*, pp. 720–726.

Shah, S. C. & Kusiak, A. (2004). Data mining and genetic algorithm based gene/SNP selection, *Artificial Intelligence in Medicine* 31(3): 183 – 196.

Shin, A. & Iba, H. (2003). Construction of genetic network using evolutionary algorithm and combined fitness function., *Genome Informatics* 14: 94–103.
URL: *http://view.ncbi.nlm.nih.gov/pubmed/15706524*

Sîrbu, A., Ruskin, H. J. & Crane, M. (2010a). Comparison of evolutionary algorithms in gene regulatory network model inference, *BMC Bioinformatics* 11(59).

Sîrbu, A., Ruskin, H. J. & Crane, M. (2010b). Cross-platform microarray data normalisation for regulatory network inference, *PLoS ONE* 5(11):e13822.

Sîrbu, A., Ruskin, H. J. & Crane, M. (2010c). Integrating heterogeneous gene expression data for gene regulatory network modelling, *Proceedings of the European Conference on Complex Systems*, Lisbon, Portugal. 13-17 Sept.

Sîrbu, A., Ruskin, H. J. & Crane, M. (2010d). Regulatory network modelling: Correlation for structure and parameter optimisation, *Proceedings of the IASTED Technology Conferences (ARP,RA,NANA,ComBio 2010)*, Cambridge, Massachusetts. 1-3 Nov.
URL: *http://www.actapress.com/Abstract.aspx?paperId=41573*

Souza, B. F. & Carvalho, A. P. (2005). Gene selection based on multi-class support vector machines and genetic algorithms., *Genetics and Molecular Research* 4(3): 599–607.

Spieth, C., Streichert, F., Speer, N. & Zell, A. (2005a). Clustering-based approach to identify solutions for the inference of regulatory networks, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2005)*, Vol. 1, pp. 660–667.

Spieth, C., Streichert, F., Speer, N. & Zell, A. (2005b). Multiobjective model optimization for inferring gene regulatory networks, *Proceedings of the Conference on Evolutionary Multi-Criterion Optimization, volume 3410 of Lecture Notes in Computer Science*, pp. 607–620.

Spieth, C., Streichert, F., Supper, J., Speer, N. & Zell, A. (2005). Feedback memetic algorithms for modeling gene regulatory networks, *Computational Intelligence in Bioinformatics and Computational Biology, 2005. CIBCB '05. Proceedings of the 2005 IEEE Symposium on* pp. 1–7.

Spieth, C., Streichert, F. & Zell, N. S. A. (2004). Optimizing topology and parameters of gene regulatory network models from time-series experiments, *Genetic and Evolutionary Computation - GECCO 2004*, pp. 461–470.

Spieth, C., Supper, J., Streichert, F., Speer, N. & Zell, A. (2006). JCell–a Java-based framework for inferring regulatory networks from time series data, *Bioinformatics* 22(16): 2051–2052.
URL: *http://bioinformatics.oxfordjournals.org/cgi/content/abstract/22/16/2051*

Stekel, D. (2003). *Microarray Bioinformatics*, Cambridge University Press.

Swain, M., Hunniford, T., Dubitzky, W., Mandel, J. & Palfreyman, N. (2005). Reverse-engineering gene-regulatory networks using evolutionary algorithms and grid computing, *J Clin Monit Comput* 19(4-5): 329–37.

Tan, K., Tegner, J. & Ravasi, T. (2008). Integrated approaches to uncovering transcription regulatory networks in mammalian cells, *Genomics* 91(3): 219 – 231.

Thieffry, D. (1999). From global expression data to gene networks, *BioEssays* 21: 895–899.

Tominaga, D., Okamoto, M., Maki, Y., Watanabe, S. & Eguchi, Y. (1999). Nonlinear numerical optimization technique based on a genetic algorithm for inverse problems: Towards the inference of genetic networks, *GCB99 German Conference on Bioinformatics*, pp. 101–111.

Vohradsky, J. (2001). Neural network model of gene expression, *The FASEB Journal* 15: 846–854.

Wahde, M. & Hertz, J. (2000). Coarse-grained reverse engineering of genetic regulatory networks, *Biosystems* 55(1-3): 129–136.
URL: *http://dx.doi.org/10.1016/S0303-2647(99)00090-8*

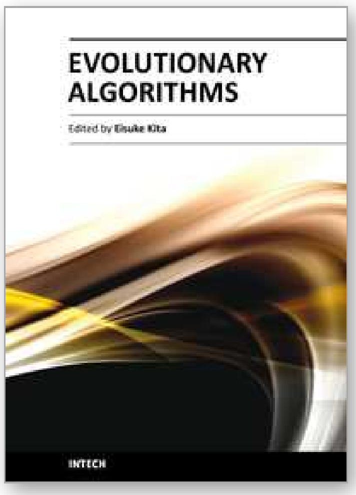
Xu, R., Venayagamoorthy, G. K. & Donald C. Wunsch, I. (2007). Modeling of gene regulatory networks with hybrid differential evolution and particle swarm optimization, *Neural Networks* 20(8): 917–927.

Zhu, Z., Jia, S. & Ji, Z. (2010). Towards a memetic feature selection paradigm, *Computational Intelligence Magazine* 5(2): 41–53.

Zhu, Z. & Ong, Y.-S. (2007). Memetic algorithms for feature selection on microarray data, *in* D. Liu, S. Fei, Z.-G. Hou, H. Zhang & C. Sun (eds), *Advances in Neural Networks Ű ISNN 2007*, Vol. 4491 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 1327–1335.

Zhu, Z., Ong, Y.-S. & Dash, M. (2007). Markov blanket-embedded genetic algorithm for gene selection, *Pattern Recognition* 40(11): 3236–3248.

Zhu, Z., Ong, Y.-S. & Zurada, J. (2010). Identification of full and partial class relevant genes, *Computational Biology and Bioinformatics, IEEE/ACM Transactions on* 7(2): 263 –277.



Evolutionary Algorithms
Edited by Prof. Eisuke Kita

ISBN 978-953-307-171-8
Hard cover, 584 pages
Publisher InTech
Published online 26, April, 2011
Published in print edition April, 2011