BIOINFORMATICS ORIGINAL ARTICLE Vol. 27 no. 6 2011, pages 791–796
doi:10.1093/bioinformatics/btr036

Structural bioinformatics

Advance Access publication January 22, 2011

Computational inference of grammars for larger-than-gene structures from annotated gene sequences

Guy Tsafnat[1,*], Jaron Schaeffer[1], Andrew Clayphan[1], Jon R. Iredell[2], Sally R. Partridge[2] and Enrico Coiera[1]

[1]Centre for Health Informatics, Australian Institute of Health Innovation, University of New South Wales and [2]Centre for Infectious Diseases and Microbiology, University of Sydney, Sydney, Australia

Associate Editor: Ivo Hofacker

ABSTRACT

Motivation: Larger than gene structures (LGS) are DNA segments that include at least one gene and often other segments such as inverted repeats and gene promoters. Mobile genetic elements (MGE) such as integrons are LGS that play an important role in horizontal gene transfer, primarily in Gram-negative organisms. Known LGS have a profound effect on organism virulence, antibiotic resistance and other properties of the organism due to the number of genes involved. Expert-compiled grammars have been shown to be an effective computational representation of LGS, well suited to automating annotation, and supporting *de novo* gene discovery. However, development of LGS grammars by experts is labour intensive and restricted to known LGS.

Objectives: This study uses computational grammar inference methods to automate LGS discovery. We compare the ability of six algorithms to infer LGS grammars from DNA sequences annotated with genes and other short sequences. We compared the predictive power of learned grammars against an expert-developed grammar for gene cassette arrays found in Class 1, 2 and 3 integrons, which are modular LGS containing up to 9 of about 240 cassette types.

Results: Using a Bayesian generalization algorithm our inferred grammar was able to predict $>95\%$ of MGE structures in a corpus of 1760 sequences obtained from Genbank (*F*-score 75%). Even with 100% noise added to the training and test sets, we obtained an *F*-score of 68%, indicating that the method is robust and has the potential to predict *de novo* LGS structures when the underlying gene features are known.

Availability: http://www2.chi.unsw.edu.au/attacca.

Contact: guyt@unsw.edu.au

Received on April 16, 2010; revised on January 16, 2011; accepted on January 18, 2011

1 INTRODUCTION

The study of DNA has particularly focused on regions that code RNA (primarily genes) and regions that interact with proteins such as transcription factor binding sites. In 1961, Jacob and Monod (1961) published the first example of a gene regulation mechanism that exploits the position of genes to simultaneously control the expression of genes that are part of the same metabolic pathway. Thomas and Nielsen (2005) list 12 classes of larger than gene structures (LGS) that can transfer regions of DNA containing one or more genes between DNA molecules in a cell or, through conjugation, between organisms. Many examples of each class of these mobile genetic elements (MGEs) are found in GenBank (Partridge *et al*., 2009; Siguier *et al*., 2006). These MGEs have all been discovered after initial observations of unexpected phenomena. Integrons, for example, were first identified when manual observations found a similar sequence pattern (restriction enzyme digestion sites) repeatedly flanking a variety of quite different antibiotic resistance genes (Stokes and Hall, 1989). The development of a systematic method for making such observations would potentially reveal new structures and new mechanisms for gene regulation and transposition.

Methods of computational support for MGE discovery typically focus on a subset of MGEs called genomic islands (Langille *et al*., 2010). These methods include:

- Methods that use sequence similarity, either through whole-genome alignment (Darling *et al*., 2004), or clusters of genes (Bohnebeck *et al*., 2008). These methods depend on having similar genomes that differ by inserted genomic islands. They cannot discover the structure of the inserted sequence.
- Methods that rely the on similarity of genomic sequence properties of such as GC content (Ou *et al*., 2007) or oligonucleotide frequencies (Tsirigos and Rigoutsos, 2005). These methods can discover insertions with limited accuracy and also do not identify sequence structure.
- Methods based upon the occurrence of direct repeats (DR) and tRNA genes to identify MGE insertions. However, as many MGE are not inserted in tRNA genes nor have DR, this approach is not generalizable (Langille *et al*., 2010).

All of these methods have been designed for bacterial chromosomal DNA and not plasmids.

An important class of tools can identify new MGE from known classes of MGE (Joss *et al*., 2009; Rowe-Magnus *et al*., 2003; Tsafnat *et al*., 2009). These methods require the creation a priori of computational models of MGE structures to permit automatic MGE annotation.

We present a method to support *de novo* discovery of LGS based upon computational grammars. Short sequences (e.g. genes, recurring motifs, DR and inverted repeats) are annotated by other methods. Our method then identifies recurring patterns in these annotations to infer new LGS and represents them as grammatical models. The grammar is then used to provide a second level of

*To whom correspondence should be addressed.

annotation that can be used to measure the accuracy of the inference method.

Computational grammars have been used to model and predict transcription binding sites (Leung *et al*., 2001), RNA folding (Rivas and Eddy, 2000) and genes (Searls, 2002). Components of LGS represented with grammars include integrons (Joss *et al*., 2009; Moura *et al*., 2009; Rowe-Magnus *et al*., 2003), insertion sequences (Siguier *et al*., 2006) and gene cassettes (Partridge *et al*., 2009).

Expert-developed LGS grammars can be used to generate accurate annotations (Tsafnat *et al*., 2009). Such automatic grammatical annotations have been used to survey antibiotic resistance bearing gene cassettes in GenBank (Partridge *et al*., 2009). In this article, we automatically construct LGS grammars from DNA sequences that have been annotated for short gene cassettes and integron-specific conserved sequences, but not LGS such as cassette arrays.

Automatic grammar inference should improve the efficiency of the grammar development process, and support computational discovery of LGS, a problem for which there are currently no methods.

The method we present here is an unsupervised learning method based on the prefix tree acceptor (PTA; Carrasco and Oncina, 1994) algorithm. This method first constructs a Markov model from a training set (Viterbi, 1967) and then merges similar subchains. The resulting grammar is simpler than an ungeneralized hidden Markov model (HMM) and can accept a wider range of inputs than present in the training set.

2 METHODS

2.1 Data

We established a database of 233 gene cassettes found in Class 1, 2 and 3 integrons (Partridge *et al*., 2009) as well as 38 sequences found between gene cassettes, which we call non-cassette insertions (NCI) and 12 known conserved sequence (CS) regions of the cassette array called the 5′-CS and 3′-CS (Class 1) and 5′- and 3′-regions in Classes 2 and 3. Each entry in this database is called a 'feature' and collectively, all 290 features are called the feature database (FDB).

We selected sequences from GenBank's nucleotides database that contained at least one feature from the FDB. We excluded sequences with the terms 'Eukaryota', 'vector' or 'synthetic' in their organism field. RefSeq entries (e.g. with accession numbers starting with 'NC_') were also excluded to reduce duplicates. We collected the resulting 1760 sequences in a sequence database (SDB).

We annotated all occurrences of each feature from the FDB found in the SDB and stored the annotations in a relational database as feature name and type and position. This led to annotation of 3593 cassettes, 88 NCI, 2737 regions and 1797 gaps (total 8215 tokens). Regions of sequences that are not annotated by this process are marked with a special gap feature which is also stored with the annotations.

The grammar was developed manually using an iterative process. It contains 21 rules and annotates the cassette arrays of Class 1, 2 and 3 integrons in either direction. In a previous study, the grammar was shown to strongly agree ($\kappa = 0.972$) with a panel of expert microbiologists. The complete grammar is given in Tsafnat *et al*. (2009).

2.2 Grammar inference

The PTA algorithm (Carrasco and Oncina, 1994) is an unsupervised relational learner that infers grammars from unlabelled example texts. PTA works by first constructing a deterministic finite-state automaton (DFSA). This stage is similar to an unsupervised HMM learner. Each edge in the DFSA represents an input token (in our case an annotated feature) and the number of times

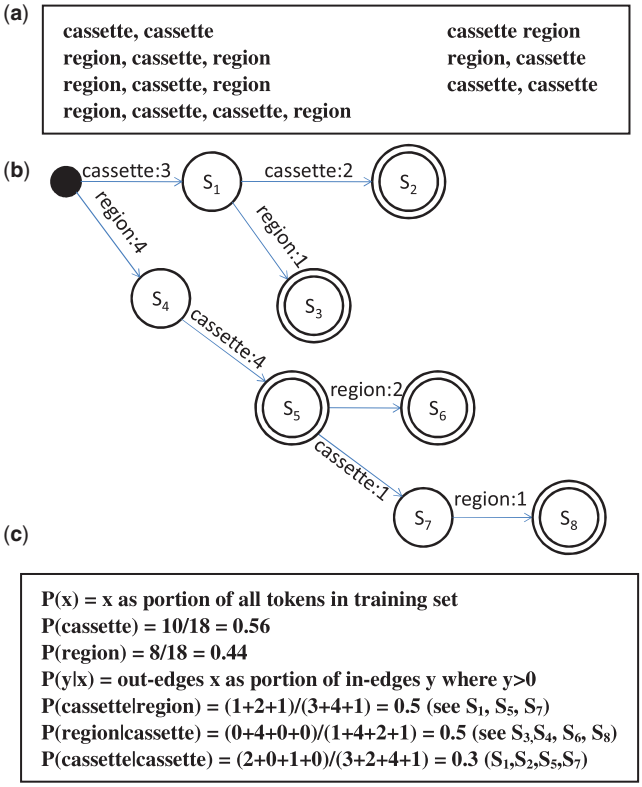

Fig. 1. An illustration of a finite state automaton (DFSA) constructed from seven training sequences (**a**) shown diagrammatically (**b**). Each edge is labelled with the semantic type of a token and a count of how many times it was traversed during the construction of the DFSA. Basic probabilities are derived from the DFSA and used to calculate transition probabilities (**c**). The nodes S_7 and S_5 score 1/1 = 1 and 2/4 = 0.5, respectively, and are thus selected for merging.

that edge was traversed when reading multiple sequences. Figure 1 shows a diagram of a DFSA built from seven example annotation sequences.

The PTA algorithm next examines the DFSA and iteratively merges states until a minimum similarity threshold is reached. Every iteration calculates the similarity of every pair of states and the most similar pair is merged if it is above the minimum threshold. This step forms loops in the DFSA graph and may also make it non-deterministic. In such cases a simple transformation is used to make it deterministic again. Loops can be thought of as generalized structures in which the sequences that form a loop form the same semantic type. By analogy to English, this allows us to consider 'sofa', 'book' and 'pencil' a single type 'noun'.

We calculate the similarity of two states in the DFSA using one of two functions that calculate a score in the range [0–1] by examining the edges of both states:

(1) A local scoring function treats the argument edges for each of the compared nodes as an N-dimensional vector: the edge's token is the axis or direction of the vector's component and each count is the magnitude of the vector along that axis. The similarity of the nodes is the cosine of the angle between the vectors. A score of 1.0 means that the feature vectors are identical, 0 means that they are orthonormal to each other, which occurs if the two feature vectors have no common axes. The score W given to each node pair i and j

with k semantic types is calculated by the formula:

$$W_{i,j} = \frac{\sum_k i_k \cdot j_k}{\sqrt{\sum_k i_k^2 + j_k^2}}$$

This method generalizes the most similar nodes under the assumption that they represent a similar pattern of tokens. However, it could also be too sensitive to noise as it considers infrequent patterns.

(2) A global scoring function that calculates the likelihood that two nodes should be merged given the overall similarities between all the nodes in the graph. The transition probability ($p_{Tx,y}$) that a token with a semantic type x is followed by one with type y (i.e. that a given state with an in-edge labelled x, will 'transition' the DFSA to an out-edge labelled y) is calculated using a Bayesian probabilistic model:

$$p_{\tau_{x,y}} = p(y|x) = \frac{p(x|y) \cdot p(y)}{p(x)}$$

where $p(x)$ is the probability of a random token in the training set to be of type x; and $p(x|y)$ is the probability of a token to be of type x, given that the token that immediately precedes it is of type y. These probabilities are calculated for the entire DFSA simultaneously such that every node with an in-edge labelled x and an out-edge labelled y contributes to $P_{Tx,y}$. The highest $p_{Tx,y}$ is called $p_{T\text{-}MAX}$.

If $p_{T\text{-}MAX}$ is above the threshold and several nodes contribute to $p_{T\text{-}MAX}$ (Fig. 1c), we merge the two nodes that contribute the most (i.e. have highest x/y values). If only one node contributes to $p_{T\text{-}MAX}$ then the we use next highest $p_{Tx,y}$ provided it is too above the threshold. The global method considers only the most common transitions (patterns), and thus ignores incidental similarities between nodes that may be caused by noisy data.

We assumed that LGS may occur in either direction in the data and so we tested each scoring method using three sets of arguments: the input edges of the compared states (Prefix), the output edges (Postfix) or both kinds of edges (Context). Hence, we compared six unique generalization methods.

In machine learning terms, this algorithm is classified as unsupervised as it attempts to elucidate structures and these structures are not annotated in the training set. We note that while the training set is annotated, it is only annotated with low-level entities, learning of which is not being attempted here.

2.3 Structure prediction

The context-free grammars generated from the PTA are expressed in Chomsky Normal Form (CNF; Chomsky, 1957) and hence produce binary parse trees (see lower parse tree in Fig. 2). The generated grammars are used to annotate the test set resulting in a parse tree for each GenBank sequence. Each internal (i.e. non-leaf) node in the parse tree represents a potential LGS, but some represent coincidental and meaningless associations. We predict an important LGS as those that are less likely to occur frequently in the SDB by chance. To distinguish such LGS we calculated the weight W of each symbol s according to the formula: $W_s = \Sigma_i L_{S,i}$ where L is the number of leaves in the parse tree of a given occurrence i of s. This formula, gives preference for long structures (i.e. with many leaves) that occur frequently in the corpus over short ones that are more likely to have occurred together by chance. Thus, the sequence in Figure 2 contributes 1 to the score W_{S1} as there is one feature below $S1$, and 9 to the score W_{S3} as there are 4 features in the subtree of one $S3$ symbol and 5 in the other. The two highest scoring symbols are designated as structure-prediction symbols, meaning that the algorithm predicts that the features in their subtrees form an important LGS.

We measured the accuracy of predictions compared with existing gold standard cassette array annotations established previously (Tsafnat *et al.*, 2009) in two ways. First, we compared the leaf nodes in the parse trees of each of the predicted structures with the leaves of the parse trees in the gold standard (Fig. 2). We also compare whether leaf tokens classified as belonging to a LGS in the gold standard cassette array (CassArray) also belongs to structure predicting symbols (e.g. S4 in Fig. 2). A leaf token appearing in both is labelled a true positive (TP). False positives (FP) are leaf tokens of predicted structures that are not in a subtree of a CassArray. Features of CassArray tokens that are not in structure predicting symbols are false negatives (FN). All other annotated features belong to neither of the groups and are hence true negatives (TN). This method provides an evaluation of how well the method predicts whole structures. It is, in a sense, more stringent than the proportionate method as it only accepts whole structures, but it is more lenient in that it accepts gaps before an after the structure.

We also measured structure matches where a structure predicting token matches the position and length (ignoring gaps before or after the structure) of a CassArray in the gold standard (TP). Nested structure predictions (i.e. when a structure predicting symbol occurs in a subtree of another structure predicting symbol) are ignored. All other structure predicting symbols are considered FP even if they overlap a CassArray symbol. Non-structure predicting symbols are counted as FN if they exactly match CassArray symbols or TN otherwise. This evaluation measures prediction of the structure itself, as well as the region it occurs in.

2.4 Generalization method comparison and parameter optimization

We compared the performance of all six generalization methods with an HMM and an algorithm that randomly merges states. We ran each experiment 30 times with different random training sets consisting of 10% ($N = 176$) of sequences.

We varied the minimum similarity threshold between 0 and 1 to find the optimum for each algorithm. We used the optimal threshold to calculate the performance of the best local and global generalization methods, given a training set of between 5 and 50% of the SDB.

2.5 Noise tolerance

To see how the method performs with noisy datasets, we gradually added features not normally found in cassette arrays to the anno-annotated SDB. We randomly selected annotations from 101 features associated with antibiotic resistance, which include antibiotic resistance genes (not found in gene cassettes), insertion sequences, Tn3-like transposons and inverted repeats of Class 1, 2 and 3 integrons. To the 8215 gene-cassette annotations used in the experiments above, we added an amount equivalent to between 5% (411) and 100% (8215) of extra annotations and adjusted gap tokens accordingly.

3 RESULTS

3.1 Comparing generalization methods

Results from repeating each experiment 30 times with different randomly selected training sets, are reported in Table 1 (proportionate) and Table 2 (whole structure) relative to HMMs and a generalizer that randomly merges DFSA states.

All six generalization methods outperform the HMM and random generalizer. The tables report average sensitivity ($\mathrm{Sn} = \mathrm{TP}/(\mathrm{TP}+\mathrm{FN})$), specificity ($\mathrm{Sp} = \mathrm{TN}/(\mathrm{TN}+\mathrm{FP})$), F-scores ($F = 2 \times \mathrm{Sn} \times \mathrm{PPV}/(\mathrm{Sn}+\mathrm{PPV})$), positive predictive value $\mathrm{PPV} = \mathrm{TP}/(\mathrm{TP}+\mathrm{FP})$ and area under the receiver operating curve (TP rate versus FP rate).

We measured the performance of each generalization methods for various thresholds (Fig. 3). The global methods performed best with a threshold of 0.3. The local generalization methods did not vary much in performance except between zero and non-zero thresholds. This is because similarities between states using these methods tend to be binary: either two states have very similar in-edges or very dissimilar ones.

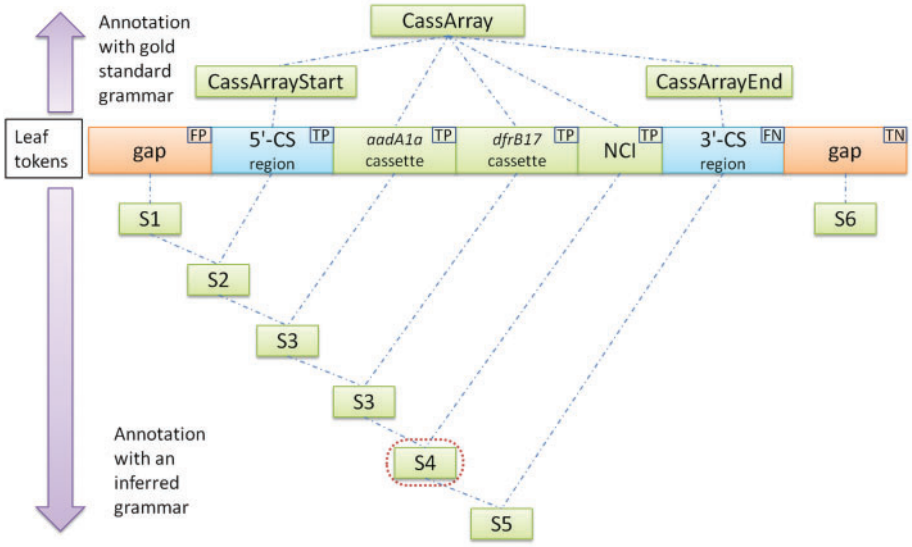

Fig. 2. A hypothetical sequence showing a cassette array annotated with the expert-developed grammar (top) and with an inferred grammar (bottom). Agreement between the two grammars is measured as overlap of the annotated structure, in this case symbols in the S4 tree and the CassArray tree.

Table 1. Evaluation of various grammar inference algorithms using 30 random subsamples of 10% of the corpus ($N = 176$)

Generalization category	Sensitivity	Specificity	F-score ±1 SD	AUC	Best threshold	Average no. of rules
Local						
(prefix)	99.9	18.5	72.6 ± 0.26	0.59	0.1	6.0
(postfix)	65.9	67.5	69.6 ± 7.2	0.70	1.0	51.5
(context)	58.7	67.2	61.7 ± 7.5	0.64	0.9	51.7
Global						
(prefix)	96.0	23.6	74.8 ± 1.96	0.65	0.3	13.1
(postfix)	96.4	20.4	74.0 ± 1.4	0.62	0.3	10.2
(context)	96.4	20.4	75.0 ± 1.1	0.64	0.3	10.6
HMM	45.6	70.8	53.26 ± 3.7	0.62	N/A	150.6
Random						
Generalizer	22.0	82.4	27.8 ± 22.5	0.52	N/A	72.1

Each result is based on the best achieved F-score for the algorithm, the threshold used to obtain it and the number of rules it generated and the area under the recipient operating curve (AUC).

Table 2. Evaluation of various grammar inference algorithms from Table 1

Generalization category	Sensitivity	Specificity	F-score ±1 SD	AUC
Local				
(prefix)	11.7	19.3	41.9 ± 0.1	0.63
(postfix)	24.0	80.1	39.4 ± 21	0.65
(context)	5.3	73.3	10.7 ± 22	0.48
Global				
(prefix)	16.3	46.5	49.3 ± 4.5	0.69
(postfix)	13.6	31.1	44.9 ± 3.3	0.65
(context)	14.9	39.2	47.1 ± 2.5	0.67
HMM	0	76.0	0 ± 0	0.43
Random				
Generalize	0.5	74.3	1.1 ± 4.0	0.43

Results are based on matched CassArray symbols in the gold standard with structure-predicting symbols in the generated grammars.

3.2 Training set size

We varied the size of the training set for Local/Prefix and the Global/Context methods between 2.5 and 50%. We used 1.0 and 0.3, respectively, as the minimum similarity thresholds for state merging. We found that in both cases, a 5% training set is sufficient for achieving a very similar F-score to that achieved using a 50% training set (Fig. 4).

3.3 Noise tolerance

We measured the performance of the global generalize using 30% transition probability threshold and 10% ($N = 176$ sequences) training set randomly chosen 30 times (Fig. 5). As expected, added noise reduces the quality of the inferred grammars, but even when half the data is noise, performance is still high with F-score of 0.68.

4 DISCUSSION

The results indicate that, at least in our domain, inferring grammars for LGS can be achieved with a moderate training set size. Selecting sequences for the SDB from GenBank may have worked in our favour as GenBank's bias is likely to over-represent Class 1 integrons compared with the general bacterial population. This bias is unavoidable and may work against, not for, *de novo* structure discovery. However, the relatively small training set needed to identify a recurring structure and the high tolerance to noise are encouraging and mean that despite this bias, new structures may yet be found in GenBank. Further experiments are required to test algorithm performance on sequences from natural populations. In this article, we have set a framework for such evaluation.

We have used two methods to evaluate the ability of the generated grammars to identify structures correctly. Proportional matching of the grammar accounts for structures that were partially discovered.

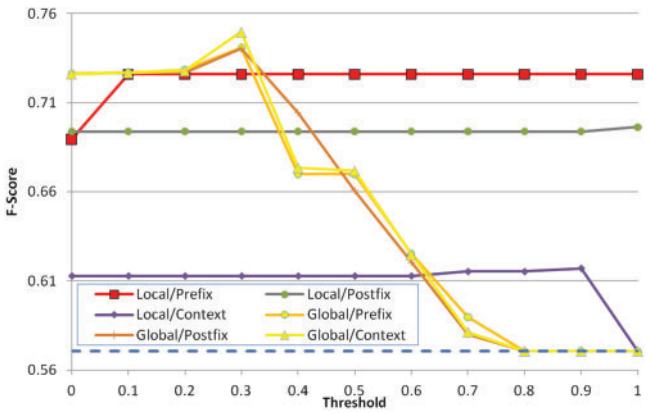

Fig. 3. The performance (F-score) of the six generalization methods and the HMM with varying thresholds. The best results for each are given in Table 1. The dashed line is HMM F-score.

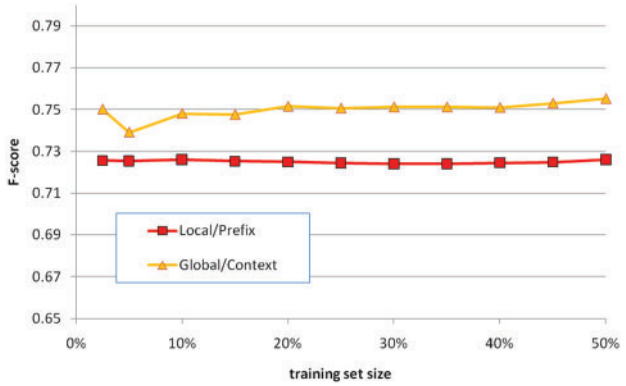

Fig. 4. The F-score performance of each of the best local and global generalization methods given for various training set sizes.

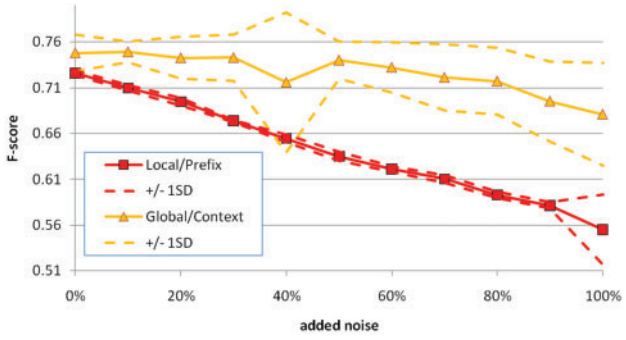

Fig. 5. The effect of noise on the performance of the best local and global generalization methods. The X-axis is noise added as percentage of the number of annotations in the clean dataset. Dashed lines show one standard deviation from average of 30 random subsamples of 10% training sets.

For example, when a sequence of five tokens (leaves) has a gold standard CassArray annotation but only four were annotated with a structure predicting token. In this case, the annotation itself is not exactly right but the correct region was indeed identified. We have shown that our method outperforms HMMs using this evaluation.

We also report a measure based upon whole structure identification. HMMs predict arrays very poorly, in particular when negative predictions are assigned the same weight as positive ones (F-score). A possible explanation to HMMs' poor performance is the small training set (10%) and we have shown that adding generalization improves performance.

Among horizontal gene transfer mechanisms, gene cassette arrays are the most modular with the highest variability in the type and number of antibiotic resistance genes. We therefore believe that our algorithm will be able to identify other structures such as transposons and insertion sequences much more accurately.

The noise experiments simulate a *de novo* discovery scenario where not all annotations processed by the system belong to a single structure. In these experiments, we needed to ensure integrons remain the dominant structure for a meaningful comparison with the gold standard. We thus added 'noise' annotations from several other structures to the annotation set ensuring that 'relevant' annotations are always at least 50% of the annotations our system sees leaving integrons as the dominant structure. In a *de novo* discovery scenario, we expect that annotations of parts of known structures will be removed from the annotations before the system is run so that the remaining most dominant structure is discovered.

De novo discovery of LGS using our methods relies on previously known component features (e.g. gene cassettes), but does not require prior knowledge of their relationship in a structure. Existing methods for prediction of component genes (Delcher *et al.*, 2007; Meyer *et al.*, 2008), inverted repeats (Swidan *et al.*, 2006) and other features can be used in conjunction with our method.

5 CONCLUSION

We have shown that novel LGS can be discovered in DNA sequences in which relevant features of the structure are annotated even using a relatively small training set (5%) and large amounts of noise (100%). An important aspect of this approach is that the inferred grammar represents a general model that accurately identifies structures without referring to prior knowledge about them. Hence, this method seems promising for *de novo* discovery even if only a few examples exist in the corpus.

Funding: This work was supported by separate New South Wales Health Capacity Building and Infrastructure Grants awarded to the Centre for Health Informatics and the Centre for Infectious Diseases and Microbiology and a Cure Cancer Foundation Australia Project Grant no. 430635.

Conflict of Interest: none declared.

REFERENCES

Bohnebeck,U. *et al.* (2008) MetaMine - a tool to detect and analyse gene patterns in their environmental context. *BMC Bioinformatics,* **9**, 459.

Carrasco,R.C. and Oncina,J. (eds) (1994) Learning stochastic regular grammars by means of a state merging method. In *Grammatical Inference and Applications*, Vol. 862 of *Lecture Notes in Computer Science*, Springer, Berlin/Heidelberg, pp. 139–139.

Chomsky,N. (1957) *Syntactic Structures.* The Hague, Mouton.

Darling,A.C.E. *et al.* (2004) Mauve: multiple alignment of conserved genomic sequence with rearrangements. *Genome Res.*, **14,** 1394–1403.

Delcher,A.L. *et al*. (2007) Identifying bacterial genes and endosymbiont DNA with Glimmer. *Bioinformatics*, **23**, 673–679.

Jacob,F. and Monod,J. (1961) Genetic regulatory mechanisms in the synthesis of proteins. *J. Mol. Biol.*, **3**, 318–56.

Joss,M.J. *et al*. (2009) ACID: annotation of cassette and integron data. *BMC Bioinformatics,* **10**, 118.

Langille,M.G.I. *et al.* (2010) Detecting genomic islands using bioinformatics approaches. *Nat. Rev. Microbiol.*, **8**, 373–382.

Leung,S. *et al.* (2001) Basic Gene Grammars and DNA-ChartParser for language processing of Escherichia coli promoter DNA sequences. *Bioinformatics,* **17,** 226–236.

Meyer,F. *et al*. (2008) The metagenomics RAST server - a public resource for the automatic phylogenetic and functional analysis of metagenomes. *BMC Bioinformatics,* **9**, 386.

Moura,A. *et al*. (2009) INTEGRALL: a database and search engine for integrons, integrases and gene cassettes. *Bioinformatics,* **25**, 1096–1098.

Ou,H.-Y. *et al*. (2007) MobilomeFINDER: web-based tools for in silico and experimental discovery of bacterial genomic islands. *Nucleic Acids Res*., **35**, W97–W104.

Partridge,S.R. *et al*. (2009) Gene cassettes and cassette arrays in mobile resistance integrons. *FEMS Microbiol. Rev.*, **33**, 757–784.

Rivas,E. and Eddy,S.R. (2000) The language of RNA: a formal grammar that includes pseudoknots. *Bioinformatics,* **16**, 334–340.

Rowe-Magnus,D.A. *et al*. (2003) Comparative analysis of superintegrons: engineering extensive genetic diversity in the Vibrionaceae. *Genome Res.*, **13**, 428–442.

Searls,D.B. (2002) The language of genes. *Nature,* **420**, 211–217.

Siguier,P. *et al*. (2006) ISfinder: the reference centre for bacterial insertion sequences. *Nucleic Acids Res.*, **34**, D32–D36.

Stokes,H.W. and Hall,R.M. (1989) A novel family of potentially mobile DNA elements encoding site-specific gene-integration functions: integrons. *Mol. Microbiol.*, **3**, 1669–1683.

Swidan,F. *et al.* (2006) On the repeat-annotated phylogenetic tree reconstruction problem. *J. Comput. Biol.*, **13**, 1397–1418.

Thomas,C.M. and Nielsen,K.M. (2005) Mechanisms of, and barriers to, horizontal gene transfer between bacteria. *Nat. Rev. Microbiol.*, **3**, 711–721.

Tsafnat,G. *et al*. (2009) Context-driven discovery of gene cassettes in mobile integrons using a computational grammar. *BMC Bioinformatics,* **10**, 281.

Tsirigos,A. and Rigoutsos,I. (2005) A new computational method for the detection of horizontal gene transfer events. *Nucleic Acids Res.*, **33**, 922–933.

Viterbi,A. (1967) Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theory,* **13**, 260–269.