BIOINFORMATICS APPLICATIONS NOTE Vol. 20 no. 12 2004, pages 1974–1976
doi:10.1093/bioinformatics/bth179

A genetic algorithm for the detection of new cis-regulatory modules in sets of coregulated genes

Stein Aerts*, Peter Van Loo, Yves Moreau and Bart De Moor

Department of Electrical Engineering (ESAT-SCD), Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, 3001 Heverlee (Leuven), Belgium

Received on August 7, 2003; revised on December 30, 2004; accepted on January 29, 2004
Advance Access publication March 25, 2004

ABSTRACT

Summary: The implementation of a genetic algorithm is described that provides a fast method of searching for the optimal combination of transcription factor binding sites in a set of regulatory sequences.

Availability: The algorithm can be used transparently as a web service from within the Toucan software. Toucan can be accessed at http://www.esat.kuleuven.ac.be/~saerts/software/toucan.php. A standalone version of the software is available upon request.

Contact: stein.aerts@esat.kuleuven.ac.be

Microarray and other high-throughput experiments in metazoans often yield sets of coexpressed genes that might share common *cis*-regulatory modules in their promoters or enhancers. Toucan (Aerts *et al.*, 2003a) can be used to select putative regulatory regions from Ensembl (Hubbard *et al.*, 2002) and to perform so-called *cis*-regulatory analysis. This includes for example the annotation of putative transcription factor binding sites (TFBSs) and the detection of new DNA motifs. Recently, we have added a new web service that searches for the optimal combination of TFBSs in a sequence set using an A* tree search algorithm (Aerts *et al.*, 2003b). The score function that is used is essentially the sum of the log-odds scores of the best hit of each individual TF within a window of specified length L, summed over all sequences in the set. Although this method guarantees finding the optimal solution, it can be slow for certain parameter settings, for large sequence sets or for modules that contain many different transcription factors (e.g. more than five). Therefore, we have implemented another search algorithm based on genetic algorithms (GAs) that is faster and more practical. The algorithm starts with the creation of p random modules. A module is a vector that contains n_Θ position-specific probability matrices derived from TRANSFAC (Wingender *et al.*, 2000) or from other matrix collections that are available on our server. The list of modules is sorted according to the score function mentioned above, and the s highest scoring modules are retained for the reproduction step. In the reproduction step, the population grows back to size p by successive pairing and mutating of randomly selected modules. When two modules are paired, for each position in the vector one element is chosen from either of the two parents, unless this element or a similar element is already present in the child module. Each element of a child module can then be mutated according to a mutation probability ρ. After g generations, the 'fittest' module is selected as the solution.

The complexity of the algorithm is $\mathcal{O}[g(p-s)nq^{n_\Theta}]$, where n is the number of sequences in the set and q is the average number of binding sites of a transcription factor on a sequence. Figure 1A summarizes the GA procedure and Figure 1B shows visually a reproduction example.

For the technical and biological validation of the algorithm, we refer to the validation of the A* algorithm (Aerts *et al.*, 2003b). Since the GA does not guarantee optimality, the user can perform multiple runs of the GA and select only those modules that are found consistently among different runs. In order to compare GA with A* in terms of accuracy (i.e. does GA also find the optimal solution that A* finds?) and of speed, we have run the GA and the A* versions on the same set of sequences as in Aerts *et al.* (2003b). For a set of genes that are coexpressed with cyclin B2 according to a time course microarray experiment during cell cycle in human HeLa cells (Whitfield *et al.*, 2002), all human–mouse conserved sequence blocks within 10 kb upstream of the transcription start site are selected and scored with all position weight matrices of TRANSFAC using the MotifScanner. The CPU time (on a 1 GHz Pentium III processor running Red Hat Linux) taken by the GA, setting L to 100 bp and g to 100 iterations, is about 7, 10, 13 and 18 min when n_Θ is set to 4, 5, 6 and 7, respectively. The time required for A* increases more dramatically with n_Θ. For $n_\Theta = 4$, A* takes about 30 min, and for $n_\Theta = 5$ it takes between 5 h and 3 days, depending on the dataset and on L. $n_\Theta > 5$ was not feasible for this particular dataset, neither in time nor in memory.

*To whom correspondence should be addressed.

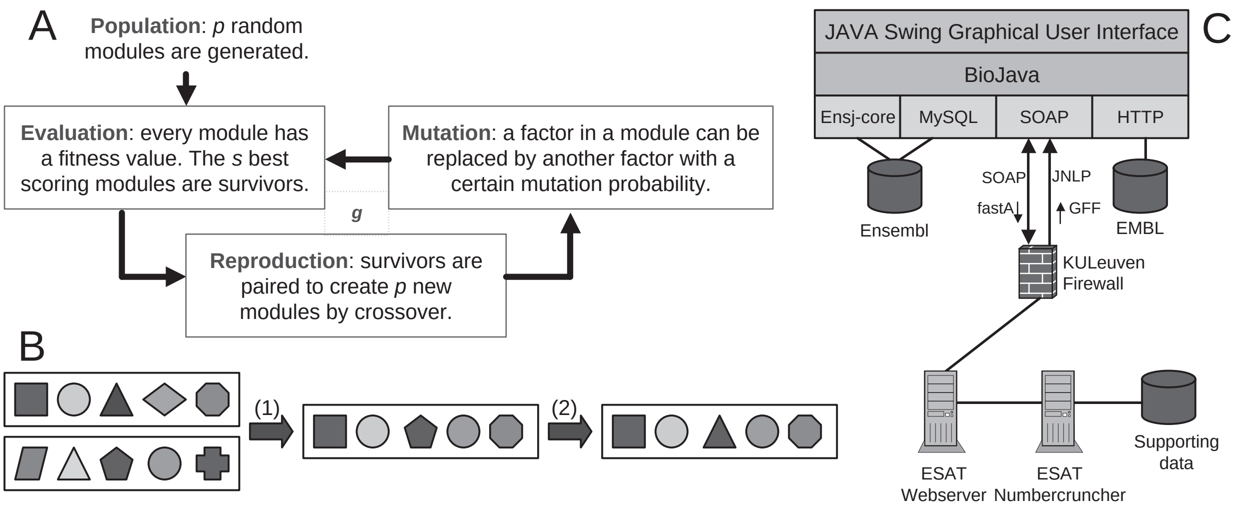
A
Population: p random modules are generated.
Evaluation: every module has a fitness value. The s best scoring modules are survivors.
Mutation: a factor in a module can be replaced by another factor with a certain mutation probability.
g
Reproduction: survivors are paired to create p new modules by crossover.
B
(1)
(2)
C
JAVA Swing Graphical User Interface
BioJava
Ensj-core
MySQL
SOAP
HTTP
Ensembl
SOAP
JNLP
fastA
GFF
EMBL
KULeuven Firewall
ESAT Webserver
ESAT Numbercruncher
Supporting data

Fig. 1. (**A**) Procedure of the GA; g is the number of generations. (**B**) Example of the generation of child modules by pairing (1) and mutations (2). Each geometrical figure represents a transcription factor. (**C**) The Toucan software environment showing the use of BioJava, Ensj-core and SOAP web services.

The maximum scores of three GA runs with 100 iterations are, for $n_\Theta = 3, 4, 5$, exactly the same (and thus the optimal module is found) as in A*. Although we have no results of A* for $n_\Theta > 5$, the results of GA for larger n_Θs show the same scores in multiple runs of the GA (e.g. in two out of three runs), and therefore, these can be assumed to be the optimal scores. In conclusion, the GA version of the ModuleSearcher is capable of finding the optimal combination of binding sites without a limitation on the number of sites and within a fraction of the time that A* needs.

A newly found module should be validated *in silico* by screening the full genome of the species that was used. For this purpose, several methods have been published recently that take the individual matrices of a module as input and that return putative hits with a certain statistical significance: COMET (Frith *et al*., 2002), MSCAN (Johansson *et al*., 2003), Stubb (Sinha *et al*., 2003), CREME (Sharan *et al*., 2003), MCAST (Bailey and Noble, 2003) and ModuleScanner (Aerts *et al*., 2003b). A module that was found in the ‘training set’ by using the ModuleSearcher (either the A* or the GA version) can be retained for experimental validation in case (1) multiple top-scoring genes found in the genome-scan overlap with the genes of the training set; and (2) the top-scoring genes are functionally coherent and related to the function of the genes in the training set. The latter can be investigated by comparing the over-represented Gene Ontology annotations of both gene sets, using tools like FatiGO (http://fatigo.bioinfo.cnio.es/), GOMiner (Zeeberg *et al*., 2003), EASE (http://david.niaid.nih.gov/david/ease.htm) or GO4G (Coessens *et al*., 2003).

The ModuleSearcher is available within Toucan. This is a Java application that can be launched directly from our web site using Java Web Start. Behind the user interface, we have made extensive use of the BioJava library for all sequence and annotation actions. The bottom layer of the application serves two goals: data access classes and web service client classes. The Ensj-core library of Ensembl is used to retrieve genes, transcripts and annotations either from the public Ensembl database or from a local Ensembl installation. Via the MySQL classes, direct queries to the Ensembl MySQL database are also possible. The Apache SOAP (simple object access protocol) implementation is used to send requests in XML format to services running on our servers. For most services, a fastA formatted string together with some parameters of the algorithm is sent to the service (running within Tomcat on Apache), and GFF formatted features are sent back to Toucan after the execution of the algorithm. For reasons of efficiency we do not run the methods on the web server itself but send RMI (remote method invocation) requests to a dedicated machine that performs all calculations. Figure 1C shows a detailed view of the design of the Toucan platform with its components and its web services.

The following web services are available currently: MotifScanner, MotifLocator, MotifSampler, AVID/VISTA (Bray *et al*., 2003), Footprinter (Blanchette and Tompa, 2002) and ModuleSearcher. To find modules, the user runs MotifScanner, MotifLocator or MotifSampler to annotate putative TFBSs and then he/she runs ModuleSearcher on the annotated set. A manual, tutorial, installation instructions, news list, references and a list of all available web services can be found on the application’s website.

ACKNOWLEDGEMENTS

S.A. is a research assistant at the K.U.Leuven. Y.M. is a postdoctoral researcher of the FWO-Vlaanderen, currently on leave at the Center for Biological Sequence Analysis, Danish Technical University, Lyngby, Denmark. B.D.M. is a full professor of the K.U.Leuven. Research supported by Research Council KUL, GOA-Mefisto 666, IDO; Flemish Government, FWO, projects G.0115.01, G.0240.99, G.0407.02, G.0413.03, G.0388.03, G.0229.03, ICCoS, ANMMM; AWI; IWT; STWW-Genprom, GBOU-McKnow, GBOU-SQUAD, GBOU-ANA; Belgian Federal Government, DWTC (IUAP IV-02 and IUAP V-22); EU, CAGE; ERNSI; Contract Research/agreements, VIB.

REFERENCES

Aerts,S., Thijs,G., Coessens,B., Staes,M., Moreau,Y. and De Moor,B. (2003a) Toucan: deciphering the *cis*-regulatory logic of coregulated genes. *Nucleic Acids Res.*, **31**, 1753–1764.

Aerts,S., Van Loo,P., Thijs,G., Moreau,Y. and De Moor,B. (2003b) Computational detection of *cis*-regulatory modules. *Bioinformatics*, **19**(Suppl. 2), II5–II14.

Bailey,T. and Noble,W. (2003) Searching for statistically significant regulatory modules. *Bioinformatics*, **19**(Suppl. 2), II16–II25.

Blanchette,M. and Tompa,M. (2002) Discovery of regulatory elements by a computational method for phylogenetic footprinting. *Genome Res.*, **12**, 739–748.

Bray,N., Dubchak,I. and Pachter,L. (2003) AVID: a global alignment program. *Genome Res.*, **13**, 97–102.

Coessens,B., Thijs,G., Aerts,S., Marchal,K., De Smet,F., Engelen,K., Glenisson,P., Moreau,Y., Mathys,J. and De Moor.,B. (2003) INCLUSive: a web portal and service registry for microarray and regulatory sequence analysis. *Nucleic Acids Res.*, **31**, 3468–3470.

Frith,M. C., Spouge,J. L., Hansen,U. and Weng,Z. (2002) Statistical significance of clusters of motifs represented by position specific scoring matrices in nucleotide sequences. *Nucleic Acids Res.*, **30**, 3214–3224.

Hubbard,T., Barker,D., Birney,E., Cameron,G., Chen,Y., Clark,L., Cox,T., Cuff,J., Curwen,V., Down,T. *et al.* (2002) The Ensembl genome database project. *Nucleic Acids Res.*, **30**, 38–41.

Johansson,O., Alkema,W., Wasserman,W. and Lagergren,J. (2003) Identification of functional clusters of transcription factor binding motifs in genome sequences: the MSCAN algorithm. *Bioinformatics*, **19**(Suppl. 1), I169–I176.

Sharan,R., Ovcharenko,I., Ben-Hur,A. and Karp,R. (2003) CREME: a framework for identifying *cis*-regulatory modules in human-mouse conserved segments. *Bioinformatics*, **19**(Suppl. 1), I283–I291.

Sinha,S., Van Nimwegen,E. and Siggia,E. (2003) A probabilistic method to detect regulatory modules. *Bioinformatics*, **19**(Suppl. 1), I292–I301.

Whitfield,M.L., Sherlock,G., Saldanha,A.J., Murray,J.I., Ball, C.A., Alexander,K.E., Matese,J.C., Perou,C.M., Hurt,M.M., Brown,P.O. and Botstein,D. (2002) Identification of genes periodically expressed in the human cell cycle and their expression in tumors. *Mol. Biol. Cell.*, **13**, 1977–2000.

Wingender,E., Chen,X., Hehl,R., Karas,H., Liebich,I., Matys,V., Meinhardt,T., Pruss,M., Reuter,I. and Schacherer,F. (2000) TRANSFAC: an integrated system for gene expression regulation. *Nucleic Acids Res.*, **28**, 316–319.

Zeeberg,B.R., Feng,W., Wang,G., Wang,M.D., Fojo,A.T., Sunshine,M., Narasimhan,S., Kane,D.W., Reinhold,W.C., Lababidi,S. *et al.* (2003) GoMiner: a resource for biological interpretation of genomic and proteomic data. *Genome Biol.*, **4**, R28.