A Rule-Based Approach to Selecting Developmental Processes

Jan T. Kim, jtk@cmp.uea.ac.uk
School of Computing Sciences, University of East Anglia
Norwich NR4 7TJ, United Kingdom

Abstract

This paper introduces a method to generate synthetic regulatory gene networks (RGNs) using a predetermined developmental process, specified by a computer model. The candidate RGN is implanted into the developing system and a growth record comprised of gene expression levels in the system's component and the developmental steps affecting each component is obtained. Performance of an RGN is scored with an objective function based on correlations of gene expression levels and execution of developmental steps. Optimisation of the objective function yields synthetic RGNs that generate gene expression patterns which can be used to control the desired developmental process.

The concept is demonstrated using an elementary model of plant growth to perform optimisation of the numerical parameters of randomly generated synthetic RGNs, but it can be generalised to any rule based computational model of development.

1 Introduction

The complexity of development and evolution of developmental processes has intrigued bioscientists for centuries. This focus of interest has been inherited by Artificial Life, where numerous computational models and frameworks to simulate and study development and its evolution have been published [Wilson, 1989, Fleischer and Barr, 1993, Kim, 2000, Bentley and Clack, 2005].

The role of regulatory gene networks (RGNs) for organising differentiation and development based on genetic information has been recognised already by [Kauffman, 1987]. Recently, computational simulations of RGNs have attracted much renewed interest, due to progress in Molecular Systems Biology and the emergence of "Network Biology" [Barabási and Oltvai, 2004] as part of a new trend of network based methods which begins to unify research in various fields including physics, social sciences and biological disciplines. Network approaches are applied on multiple levels of biological organisation, where RGNs and ecosystems [Williams and Martinez, 2000] are perhaps the most prominent ones.

For Systems Biology, computational simulation of gene expression dynamics is important as a basis for developing and objectively evaluating methods for analysing gene expression data. Among the simulators used for researching methods for gene expression data analysis and RGN reconstruction are Gepasi [Mendes et al., 2003], `transsys` [Repsilber and Kim, 2003, Kim, 2005], and the SynTReN simulator which has recently been developed specifically for this purpose [Van den Bulcke et al., 2006].

Generating synthetic gene expression data requires biologically realistic RGN models. Various

methods for using empirical measurements of gene expression data to reconstruct RGNs have been proposed and developed [Rung et al., 2002, Basso et al., 2005]. However, using such reconstructed RGN models for evaluating gene expression analysis methods leads to a problem of circularity: The reconstruction procedure involves the methods to be analysed, and generation of the test data is thus not independent of the system to be tested.

In addition to models directly derived from empirical data ("life as it is"), the scientific object of Artificial Life includes systems that are based on the same principles as real biological systems ("life as it could be"). Computational systems of the latter type are not dependent on specific empirical data, and synthetic data generated from such computational models can therefore be used to assess RGN reconstruction methods in a way that is not affected by the circularity described above.

This report introduces an approach for generating and parameterising synthetic, computational RGN models that are capable of organising a formalised process of biological development. The computational model used to demonstrate this approach is `L-transsys` [Kim, 2001], a system for modelling plant morphogenesis based on a combination of Lindenmayer systems (L-systems) and `transsys`. RGN models that are suitable to control a predetermined process of plant development are generated by parameter fitting.

This concept requires a method to measure the similarity between the predetermined developmental process and the actual process obtained with a candidate RGN. Assessing similarity of phenotypes, or of developmental processes, is generally a difficult problem. The choice of phenotypic traits to use for comparison can be subjective, and similarity measures of 2- or 3-dimensional objects are frequently computationally expensive. The method presented here avoids these problems by focusing on the spatiotemporal sequence of developmental steps. An elementary developmental step in a computational model of development can generally be formalised as the execution of a growth rule [Kniemeyer et al., 2004]. The sequence of rule activations during a developmental process is a measurable trait that is computationally readily accessible and not dependent on subjective choice.

As an initial approach towards this concept, this paper explores the use of parameter optimisation to produce RGNs that can organise the growth of a branching structure (a classic motif in plant modelling with L-systems). The pattern of rule activation in the growth process is used to develop objective functions, and the RGN parameters are fitted to optimise these objective functions using a local search procedure.

2 Methods

2.1 Background

`transsys` [Kim, 2001] is a computational system for modelling RGNs. A `transsys` program consists of declarations of the factors (proteins or other gene products) and the genes which constitute the RGN. Fig. 1 shows an example of a `transsys` program. A `transsys` instance contains the concentration values of all factors in the program and thus describes the state of a biological unit (such as a cell or a plant component in the case of `L-transsys`).

Lindenmayer systems (L-systems) [Prusinkiewicz and Lindenmayer, 1990] represent the morphology of a plant by a string of symbols. In `L-transsys`, a symbol may be parameterised by a `transsys` instance. Plant growth is simulated by applying a set of substring replacement rules to the string representing the current plant morphology. These replacement rules represent the elementary steps of development.

```
transsys example
{
  factor A
  {
    decay: 0.1;
    diffusibility: 0.2;
  }

  factor R
  {
    decay: 0.15;
    diffusibility: 0.3;
  }

  gene agene
  {
    promoter
    {
      constitutive: 0.01;
      A: activate(0.02, 1.0);
      R: repress(0.1, 1.1);
    }
    product
    {
      default: A;
    }
  }

  gene rgene
  {
    promoter
    {
      A: activate(1.0, 10.0);
      R: repress(1.0, 1.0);
    }
    product
    {
      default: R;
    }
  }
}
```

Figure 1: An example `transsys` program consisting of two factors and two genes. The numerical parameters in this program are 0.1, 0.2 **(factor `A`)** 0.15, 0.3 **(factor `B`),** 0.01, 0.02, 1.0, 0.1, 1.1 **(gene `agene`) and** 1.0, 10.0, 1.0, 1.0 **(gene `rgene`).**

`L-transsys` is combines the `transsys` and L-systems by associating `transsys` instances with L-systems string symbols. The current `transsys` software archive, which provides more detailed technical documentation, is available from the `transsys` home page.[1]

Fig. 2 shows the `L-transsys` program of a simple branching structure. The `transsys` program `aux` provides one factor, `A`, and a gene `agene` which constitutively expresses one unit of `A` per time step. As `A` does not decay, it accumulates at a rate of one unit per time step. This factor is used to trigger execution of the rule `grow` every 5 time steps. The rule `grow` states that a `meristem` symbol in which contains at least 5 units of factor `A` is replaced by the sequence of symbols to the right of the arrow, the righthand side (RHS). The statement `transsys t: A = 0.0`, attached in parentheses to the `meristem` symbols in the RHS, specifies that the `transsys` instance of the original `meristem` symbol is copied, and the amount of factor `A` is set to 0 in the process. Fig. 3 shows the development of this simple branching structure.

The `transsys` program `aux` serves as a host into which other programs are implanted in the process of parameter optimisation (see below). Therefore, the fact that the transsys instance is copied is important, even though it is not relevant for understanding the `L-transsys` code shown in Fig. 2 (as the sole factor `A` is overwritten by the subsequent assignment in all `meristem` symbols).

Each rule in `L-transsys` is required to have a unique name (e.g. "`grow`" in the example discussed above). This allows keeping a record of all symbols generated during the growth of a simulated plant, including the `transsys` instance parameter of the symbol and the name of the rule which is activated. This allows to directly assess the correlation between gene expression levels, represented by `transsys` factor concentrations, and growth processes, represented by `L-transsys` rule executions.

[1] `http://www.cmp.uea.ac.uk/~jtk/transsys/`

```
transsys aux
{
  factor A
  {
    decay: 0.0;
    diffusibility: 0.0;
  }

  gene agene
  {
    promoter
    {
      constitutive: 1.0;
    }
    product
    {
      default: A;
    }
  }
}
```

```
lsys simplebrancher
{
  symbol meristem(aux);
  symbol shoot_piece;
  symbol left;
  symbol right;
  symbol [;
  symbol ];

  axiom meristem();

  rule grow
  {
    meristem(t) : t.A >= 5.0 -->
      [ left meristem(transsys t: A = 0.0) ]
      [ right meristem(transsys t: A = 0.0) ]
      shoot_piece meristem(transsys t: A = 0.0)
  }
}
```

Figure 2: The symbol, axiom and rule declarations of the `L-transsys` program. The symbol `meristem` is declared to be parameterised by the `transsys` program `aux`, shown left.

2.2 Objective Functions

The objective functions are designed to respond to correlations between gene expression levels and activation of growth rules. These are jointly captured in a growth record. Formally, the record of an `L-transsys` growth process is a set G of symbol records, where each symbol record is a tuple consisting of

- the time step
- the index of the symbol within the string
- the name of the symbol, denoted by s
- the name of the rule activated by the symbol, denoted by r
- the factor concentrations in the `transsys` instance parameterising the symbol, denoted by $c_1, \ldots, c_n$.

The subset of symbol records in which rule r is activated is denoted by G_r, and $\overline{G_r} = G - G_r$ denotes the subset of symbol records in which r is not activated. The set of concentration values of factor p within a set S of symbol instances is denoted by $C_p(S)$.

The set of expression levels found for factor p in symbols activating rule r is thus denoted by $C_p(G_r)$, and the expression levels of p in all other symbols are in $C_p(\overline{G_r})$. The suitability of p for determining activation of rule r can be quantified based on the disparity between these complementary sets: The greater the disparity, the better the factor is suitable for differentiating between symbols that should activate r and symbols that should not. The objective functions f_σ and f_{overlap} quantify this notion. Both objective functions are designed to evaluate to 0 if the gene product is differentially expressed such that it is can be used to accurately control activation of r. This case is referred to as perfect disparity. The objective function value is 1 if p is useless for determining activation of r.

For a factor p and a rule r, let $I_p^{r-} = [\min C_p(\overline{G_r}), \max C_p(\overline{G_r})]$ and $I_p^{r+} = [\min C_p(G_r), \max C_p(G_r)]$ and let $n_{\text{overlap}} = |\{c : c \in C_p(G), c \in I_p^{r+} \wedge c \in I_p^{r-}\}|$ be the number

of expression level values of p which are in the overlap of the intervals I_p^{r+} and I_p^{r-}. The overlap objective function is defined as

$$f_{\text{overlap}}(r,p) = \frac{n_{\text{overlap}}}{|C_p(G)|} \tag{1}$$

The overlap objective function is 0 if there is no overlap of I_p^{r+} and I_p^{r-}. This means that either $\max C_p(\overline{G_r}) < \min C_p(G_r)$ (or $\max C_p(G_r) < \min C_p(\overline{G_r})$) and that, by choosing a threshold $t_{p,r}$ in the gap between the intervals, the condition $c_p < t_{p,r}$ can fully accurately control activation of r based on the expression level of p.

For the standard deviation objective function, let c_p^{r+} denote the mean expression level and σ_p^{r+} the standard deviation of c_p in symbols activating r, and c_p^{r-} and σ_p^{r-} denote the mean expression level and standard deviation of c_p in all other symbols, respectively. The standard deviation objective function is defined by

$$f_\sigma(r,p) = \frac{(\sigma_p^{r+} + \sigma_p^{r+})/|c_p^{r+} - c_p^{r-}|}{1 + (\sigma_p^{r+} + \sigma_p^{r+})/|c_p^{r+} - c_p^{r-}|} \tag{2}$$

The rationale here is that a small spread of values (small $\sigma_p^{r+} + \sigma_p^{r+}$) and a large difference of means ($|c_p^{r+} - c_p^{r-}|$) both are indicative of an accentuated disparity.

For both objective functions, the value for a rule r is defined as as the optimum among all factors:

$$f(r,P) = \min_{p \in P} f(r,p) \tag{3}$$

where P, the proteome, denotes a set of factors. The value for the entire growth record G is given by the mean value of the per-rule objective values:

$$f(G,P) = \frac{1}{|R|} \sum_{r \in R} f(r,P) \tag{4}$$

where R denotes the set of rules in the `L-transsys` system used to generate G.

2.3 Optimisation

The parameters that are subject to optimisation are the numeric values in a `transsys` program. Various `transsys` program elements contain an arithmetic expression,[2] see Fig. 1. Each arithmetic expression may be a tree of sub-expressions, where arithmetic expressions holding numeric values are terminals of this tree. The `transsys` framework has been extended to provide methods for extracting all arithmetic expressions that represent numeric values from a `transsys` program. The values stored in these arithmetic expressions are the parameters that are subject to optimisation.

A `transsys` program is evaluated by an objective function f by first implanting the program into an the `simplebrancher` system by merging its factors and genes into the `transsys` program

[2] Unfortunately, the term "expression" is used to refer to arithmetic expressions, i.e. formulaic prescriptions to compute numeric values, as well as to gene expression, i.e. the process of synthesising the product of a gene. To avoid ambiguity, the term "expression" is qualified as either arithmetic or pertaining to genes in this paper.

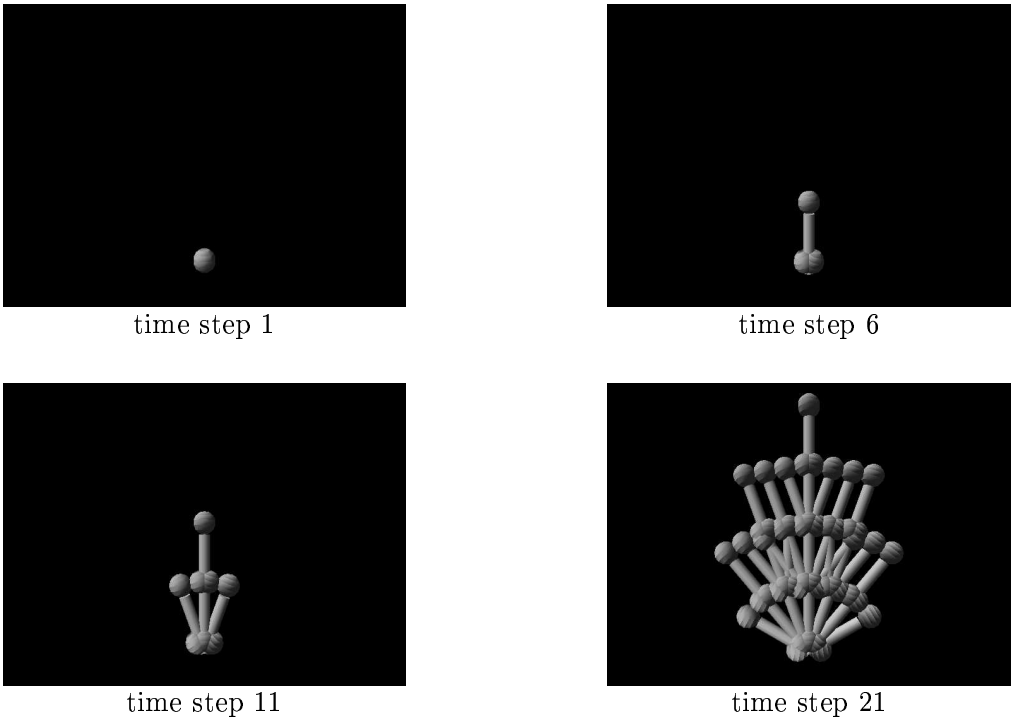


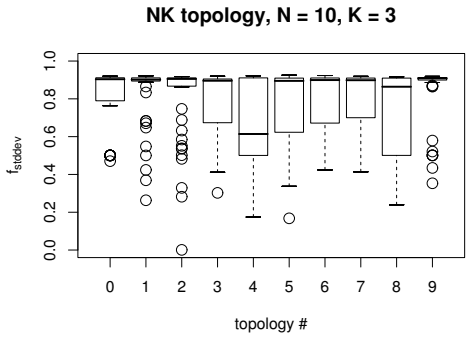
Figure 3: The developmental process used in the objective functions. The objective is to trigger the growth rule for branching every 5 time steps.

`aux`. The phenotypic growth process resulting from this system is identical to that observed with the unmodified `simplebrancher` code. After generating a growth record G of 21 time steps, the objective function value is given by $f(G, P)$, where P is the proteome of the implanted `transsys` program.

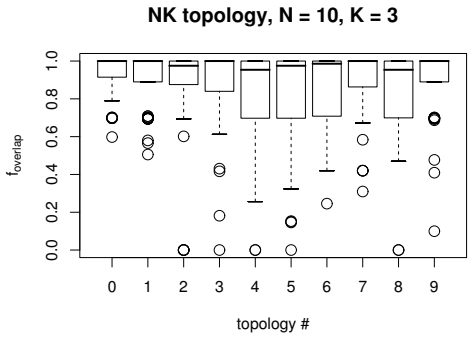
A simple, iterative local search approach is used to find a local minimum of the objective function. In each iteration, each parameter is offset by $\pm\delta$ while all other parameters are kept at their current values. Evaluations of the objective function in this δ–neighbourhood are used to estimate the local gradient. The current parameter vector is then updated by displacement along the estimated gradient. This procedure is iterated until improvement of the objective function falls below a threshold of 0.0001. For all results reported here, the offset is $\delta = 0.001$.

Initial tests showed that the random `transsys` programs could only be optimised to a limited extent using the overlap objective function. A major reason for this is that $I_p^{r+} \subset I_p^{r-}$ frequently occurs with random `transsys` programs because the rule r is activated only in relatively few symbols. If this is the case for all factors $p \in P$, $f_{\text{overlap}}(G, P) = 1$ and the gradient is flat, thus stalling the optimisation procedure. The problem local areas in which the overlap objective function is constant is further exacerbated by the fact that expression values have to move out of the overlap interval; just approaching its borders does not change the f_{overlap} result. This was one reason for choosing the relatively large offset of $\delta = 0.001$.

The standard deviation objective function is much less prone to stalling due to lack of local gradient information. Therefore, optimisation was performed sequentially by firstly optimising with f_σ and secondly subjecting the resulting `transsys` programs to optimisation with f_{overlap}. The rationale motivating this sequential optimisation procedure is that providing some degree of disparity using f_σ results in an increasing chance that the symmetric difference $I_p^{r+} \ominus I_p^{r-} \neq \emptyset$, thus providing a point of attack for optimising f_{overlap}.



NK topology, 30 edges, f_σ



NK topology, 30 edges, f_{overlap}

Figure 4: Optimisation results with the standard deviation objective function f_σ (left) and the overlap objective function f_{overlap} (right). Optimisation with f_σ was started from the value set inserted by the topology generator and from 50 random initial value sets. Optimisation of f_{overlap} was performed using the result of optimisation with f_σ as a starting point. Each box summarises the values of the objective after optimisation observed for 51 optimisation runs. Boxes depict the middle quartiles, whiskers extend to the most extreme value which is less than 1.5 times the interquartile range away from the box. Values outside the whiskers are depicted by individual points.

3 Results and Discussion

Random `transsys` programs were generated as described in [Kim, 2005]. NK graphs (in which all genes have the same in-degree K while the out-degree is subject to random variation, see [Kauffman and Weinberger, 1989]) and random graphs (Erdös–Rényi type in which both in- and out-degree are Poisson distributed) were used to generate the topology. The networks discussed here have 10 nodes (i.e. genes and gene products) were generated, with either 20, 30 or 40 edges, which, for the NK networks, corresponds to $K = 2$, $K = 3$ and $K = 4$, respectively. For each of the resulting 6 kinds of random `transsys` programs, 10 samples were generated. The numeric values in the `transsys` programs were initialised with random numbers drawn from a uniform distribution over$[0, 1[$, and the `transsys` programs were subsequently subjected to optimisation as described above. For each network topology, optimisation was performed starting from 51 different initial value sets.

Results of optimisation of transsys programs with NK topology and 30 edges are shown in Fig. 4. For all 10 topologies, the results with f_σ are subject to considerable variation, indicating a strong dependence of the outcome of optimisation on the starting parameter and thus presence of multiple local optima in the objective function. The results furthermore suggest that some `transsys` programs are more difficult to optimise than others, e.g. with topology #9, no substantial optimisation was achieved in the bulk of cases and there is only a small group of "outliers" for which $f_\sigma < 0.6$ is reached. In contrast to this, the median of f_σ is around 0.6 for topology #4.

The second stage of optimisation with f_{overlap} results in parameterisations for 5 out of the 10 topologies which produce perfect disparity. For the case of NK networks with 30 edges, a total of 10 such perfect networks are obtained (see Fig. 7).

Fig. 5 depicts a case of such perfect disparity. Factor `f0003` is expressed at levels in excess of 0.35 in all symbols that activate rule `grow`, while in all symbols that do not activate this rule, its expression level is below 0.24. For factor `f0007`, on the other hand, the range of expression levels in symbols activating `grow` is a subset of the range of expression levels found in symbols

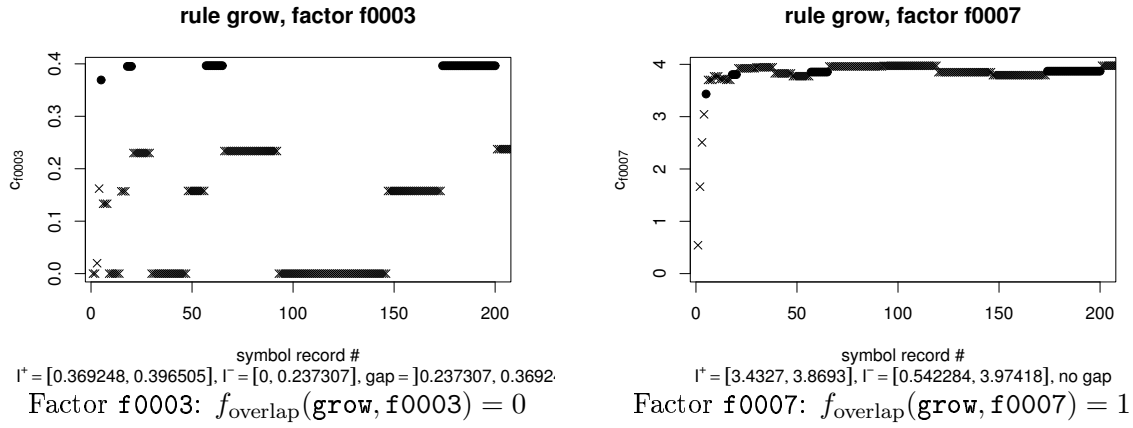


Figure 5: Plots showing the disparity of gene expression of two factors in a $N = 10, K = 3$ `transsys` program obtained by sequential optimisation. Expression levels in symbols not activating rule `grow` are shown by crosses, levels in symbols activating `grow` are shown by filled circles. The order along the horizontal axis is not important (technically, it results from the temporal sequence of symbol string sequences), the relevant difference is that for `f0003`, threshold values between 0.24 and 0.35 are suitable to determine symbols activating `grow`.

not activating that rule.

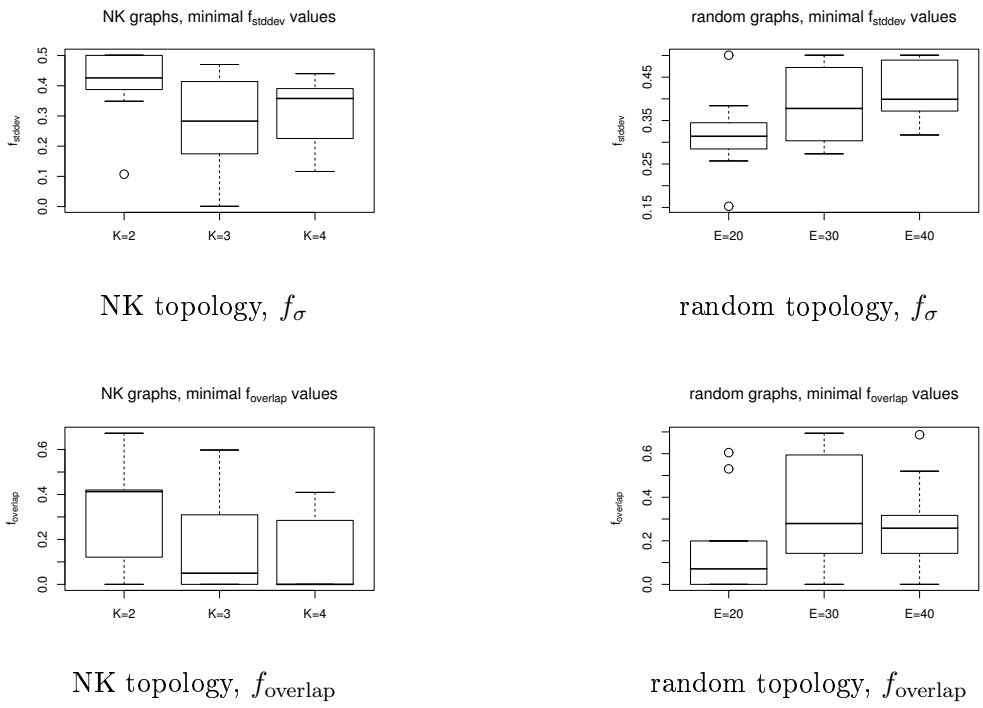
Surprisingly, analysing the complete data set obtained in this study do not show a significant effect of edge density on the performance of optimisation. However, differences are found when only the optimal results for each individual topology are considered. The results of this analysis are summarised in Fig. 6. For the NK topologies, increasing the number of edges from $K = 2$ to $K = 3$ results in some improvement of optimisation of both f_σ and f_{overlap}. For the random topologies, this trend is reversed, the best optimisation results are obtained with 20 edges and adding further edges reduces performance. The number of optimised networks that produce perfect disparity, shown in Fig. 7, appears to be correlated to the optimal values of f_σ.

The improvement seen with NK graphs with increased edge densities may be explained by the fact that more edges result in more optimisable parameters (i.e. a higher dimension of the search space), which may enable a kind of extra-dimensional bypassing [Cariani, 2002]. More specifically, the parameter optimisation procedure cannot insert new edges, but is capable of effectively removing edges by setting the maximum level of regulatory effect to 0. Therefore, the search space explored at higher edge densities implicitly contains subspaces that correspond to search spaces of networks with fewer edges. The decreased performance with high edge densities in random graphs may be due to the greater "disorder" in these graphs; while the in-degree in NK graphs is fixed to K, it is Poisson distributed in random graphs. Clearly, this is a tentative explanation and further analysis is required to understand the impact of random graph type on the structure of the objective functions and the resulting implications for the optimisation process.

4 Conclusions and Outlook

This paper presents a method to parameterise RGNs based on the correlation of gene expression with the activation pattern of growth rules during a predetermined developmental process. As a proof of concept, this method was applied to obtain `transsys` programs that are suitable to drive the development of a simple branching structure.

Developmental processes are organised based on genetic information, and consequently, computational models of development require an interface for transmitting information from the genetic



NK topology, f_σ | random topology, f_σ

NK topology, f_{overlap} | random topology, f_{overlap}

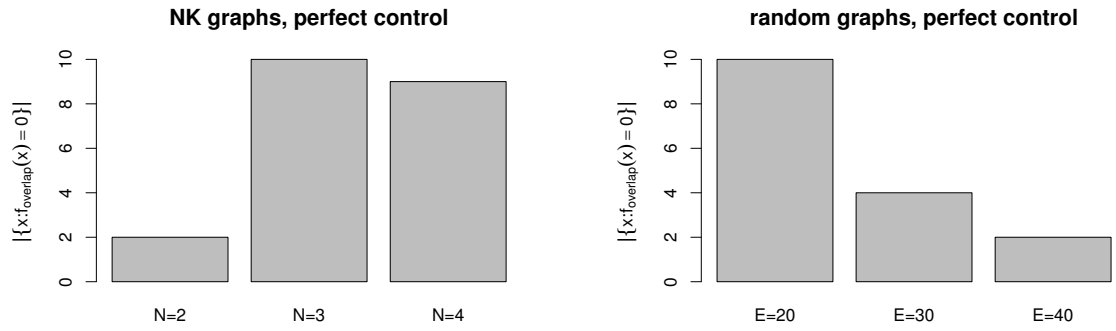
Figure 6: Boxplot of the minimal (optimal) values obtained with NK and random topologies and the f_σ and f_{overlap} objective functions.

level into the spatiotemporal process of development. Formal growth rules are a generic and fairly common element in Artificial Life models that link genetic information to development, and [Kniemeyer et al., 2004] demonstrates the potential of rule-based systems in Artificial Life modelling in general. The method presented here can therefore be applied with a large and general class of Artificial Life models. Synthetic RGNs obtained using this concept generally cannot be expected to model the corresponding molecular RGN in a sense of individual correspondences between synthetic and molecular RGN components. However, if network features exist that are required or favoured by the predetermined developmental process, they can be discovered by optimising correlations of gene expression and rule activation.

The work reported here is a step designed to enable new applications of the `transsys` framework for studying RGNs, their role in morphogenesis, and their evolution. As an immediate next step, the simulation of RGN reconstruction based on knockout mutants in [Kim, 2005] will be repeated with `transsys` programs obtained by the optimisation procedure introduced here, and controlling the `L-transsys` growth process, rather than using random `transsys` programs that receive information from the developmental process, but do not feed any information into its organisation. It will be interesting to explore whether comparing the original (wild type) symbol record with the records obtained with knockout mutants is useful to quantify the phenotypic effect of the knockout mutation.

The observed effects of edge density on optimisation performance with the two different types of network topology appear to be contradictory and are currently not satisfactorily explained. This highlights the currently limited understanding of the relationship between (static) network structure and regulatory dynamics. It might be interesting to use centrality measures [Koschützki and Schreiber, 2004] to further investigate this matter.

The objective functions introduced in this paper are multimodal and their optimisation is chal-



NK topologies random topologies

Figure 7: Barplots of the number of optimisations yielding $f_{\text{overlap}}(G, P) = 0$, i.e. providing a factor that can fully accurately control the simulated plant's growth.

lenging, motivating the use of more advanced optimisation methods, such as Simulated Annealing. Following Artificial Life traditions, it is particularly attractive to invoke evolution for optimisation. To enable such approaches, a genome interpreter mechanism (see [Kim, 2000]), based on concepts similar to [Reil, 1999, Banzhaf, 2003, Schneider, 2000], has recently been added to the `transsys` framework. Mutating sequences that encode `transsys` programs may not only alter parameters but also introduce or delete genes and regulatory links. Therefore, this mechanism enables extension of optimisation to include the network structure in addition to its numeric parameters.

The results presented here are based on a very simple developmental process, and it can be expected that more complex developmental processes, involving more rules and comprised of more time steps and larger growth records, give rise to more demanding objective functions. Their optimisation can be tackled using an incremental approach in which the temporal length of the developmental process used in the objective function is gradually increased. This would allow the optimisation process to first achieve a good performance on the earlier stages of development without penalties for mistakes in the later stages, which might prevent the objective function from adequately reflecting success in early development. While this may render the optimisation procedure susceptible to committing itself to unfavourable local optima, exploring this approach using evolutionary methods for optimisation is attractive from a biological perspective. Biological development may indeed evolve in a comparable incremental way, which may be one of the roots of Haeckel's "biogenetic law" [Theißen and Saedler, 1995].

References

[Banzhaf, 2003] Banzhaf, W. (2003). On the dynamics of an artificial regulatory network. In Banzhaf, W., Christaller, T., Dittrich, P., Kim, J. T., and Ziegler, J., editors, *Advances in Artificial Life (ECAL 2003)*, volume 2801 of *Lecture Notes in Artificial Intelligence*, pages 217–227, Berlin Heidelberg. Springer Verlag.

[Barabási and Oltvai, 2004] Barabási, A.-L. and Oltvai, Z. N. (2004). Network biology: Understanding the cell's functional organization. *Nature Reviews Genetics*, 5:101–113.

[Basso et al., 2005] Basso, K., Margolin, Adam A. anad Stolovitzky, G., Klein, U., Dalla-Favera, R., and Califano, A. (2005). Reverse engineering of regulatory networks in human B cells. *Nature Genetics*, 37:382–390.

[Bentley and Clack, 2005] Bentley, K. and Clack, C. (2005). Morphological plasticity: Environmentally driven morphogenesis. In Capcarrere, M., Freitas, A. A., Bentley, P. J., Johnson, C. G., and Timmis, J., editors, *Advances in Artificial Life (ECAL 2005)*, volume 3630 of *Lecture Notes in Artificial Intelligence*, pages 118–127, Berlin Heidelberg. Springer Verlag.

[Cariani, 2002] Cariani, P. A. (2002). Extradimensional bypass. *BioSystems*, 64:47–53.

[Fleischer and Barr, 1993] Fleischer, K. and Barr, A. H. (1993). A simulation testbed for the study of multicellular development: The multiple mechanisms of morphogenesis. In Langton, C. G., editor, *Artificial Life III*, pages 389–416, Redwood City, CA. Addison-Wesley.

[Kauffman, 1987] Kauffman, S. A. (1987). Developmental logic and its evolution. *BioEssays*, 6:82–87.

[Kauffman and Weinberger, 1989] Kauffman, S. A. and Weinberger, E. W. (1989). The NK model of rugged fitness landscapes and its application to maturation of the immune response. *J. Theor. Biol.*, 141:211–245.

[Kim, 2000] Kim, J. T. (2000). Lindevol: Artificial models for natural plant evolution. *Künstliche Intelligenz*, 1/2000:26–32.

[Kim, 2001] Kim, J. T. (2001). `transsys`: A generic formalism for modelling regulatory networks in morphogenesis. In Kelemen, J. and Sosík, P., editors, *Advances in Artificial Life (ECAL 2001)*, volume 2159 of *Lecture Notes in Artificial Intelligence*, pages 242–251, Berlin Heidelberg. Springer Verlag.

[Kim, 2005] Kim, J. T. (2005). Effects of spatial growth on gene expression dynamics and on regulatory network reconstruction. In Capcarrere, M., Freitas, A. A., Bentley, P. J., Johnson, C. G., and Timmis, J., editors, *Advances in Artificial Life (ECAL 2005)*, volume 3630 of *Lecture Notes in Artificial Intelligence*, pages 825–834, Berlin Heidelberg. Springer Verlag.

[Kniemeyer et al., 2004] Kniemeyer, O., Buck-Sorlin, G. H., and Kurth, W. (2004). A graph grammar approach to artificial life. *Artificial Life*, 10(4):413–431.

[Koschützki and Schreiber, 2004] Koschützki, D. and Schreiber, F. (2004). Comparison of centralities for biological networks. In Giegerich, R. and Stoye, J., editors, *Proceedings of the German Conference on Bioinformatics (GCB 2004)*, pages 199–206, Berlin Heidelberg. Springer Verlag.

[Mendes et al., 2003] Mendes, P., Sha, W., and Ye, K. (2003). Artificial gene networks for objective comparison of analysis algorithms. *Bioinformatics*, 19:ii122–ii129.

[Prusinkiewicz and Lindenmayer, 1990] Prusinkiewicz, P. and Lindenmayer, A. (1990). *The Algorithmic Beauty of Plants*. Springer-Verlag, New York.

[Reil, 1999] Reil, T. (1999). Dynamics of gene expression in an artificial genome – implications for biological and artificial ontogeny. In Floreano, D., Nicoud, J.-D., and Mondada, F., editors, *Advances in Artificial Life*, Lecture Notes in Artificial Intelligence, pages 457–466, Berlin Heidelberg. Springer-Verlag.

[Repsilber and Kim, 2003] Repsilber, D. and Kim, J. T. (2003). Developing and testing methods for microarray data analysis using an artificial life framework. In Banzhaf, W., Christaller, T., Dittrich, P., Kim, J. T., and Ziegler, J., editors, *Advances in Artificial Life (ECAL 2003)*, volume 2801 of *Lecture Notes in Artificial Intelligence*, pages 686–695, Berlin Heidelberg. Springer Verlag.

[Rung et al., 2002] Rung, J., Schlitt, T., Brazma, A., Freivalds, K., and Vilo, J. (2002). Building and analysing genome-wide gene disruption networks. *Bioinformatics*, 18:S202–S210.

[Schneider, 2000] Schneider, T. D. (2000). Evolution of biological information. *Nucleic Acids Research*, 28:2794–2799.

[Theißen and Saedler, 1995] Theißen, G. and Saedler, H. (1995). MADS-box genes in plant ontogeny and phylogeny: Haeckel's 'biogenetic law' revisited. *Current Opinion in Genetics and Development*, 5:628–639.

[Van den Bulcke et al., 2006] Van den Bulcke, T., Van Leemput, K., Naudts, B., van Remortel, P., Ma, H., Veschoren, A., De Moor, B., and Marchal, K. (2006). SynTReN: A generator of synthetic gene expression data for design and analysis of structure learning algorithms. *BMC Bioinformatics*, 7:43.

[Williams and Martinez, 2000] Williams, R. J. and Martinez, N. D. (2000). Simple rules yield comples food webs. *Nature*, 404:180–183.

[Wilson, 1989] Wilson, S. W. (1989). The genetic algorithm and simulated evolution. In Langton, C. G., editor, *Artificial Life I*, pages 156–166, Redwood City, CA. Addison-Wesley.