Variable neighborhood search for reverse engineering of gene regulatory networks

Charles Nicholson a,⇑, Leslie Goodwin a, Corey Clark b

a School of Industrial and Systems Engineering, University of Oklahoma, Norman, OK, United States
b Guidhall, Southern Methodist University, Dallas, TX, United States

A R T I C L E   I N F O
Article history:
Received 4 December 2015
Revised 16 November 2016
Accepted 27 November 2016
Available online 2 December 2016

Keywords:
Gene regulatory networks
Reverse engineering
Variable neighborhood search

A B S T R A C T
A new search heuristic, Divided Neighborhood Exploration Search, designed to be used with inference algorithms such as Bayesian networks to improve on the reverse engineering of gene regulatory networks is presented. The approach systematically moves through the search space to find topologies representative of gene regulatory networks that are more likely to explain microarray data. In empirical testing it is demonstrated that the novel method is superior to the widely employed greedy search techniques in both the quality of the inferred networks and computational time.

1. Background

A gene regulatory network (GRN) is a collection of genes, regulators, and regulatory connections that govern expression levels [1]. Analysis of GRNs has become essential for better understanding cellular systems because it provides insight into which genes control the activation of others [2,3]. The network topology has various interpretations in literature: the nodes in the GRN may represent genes or their protein products, the undirected edges between nodes may indicate genes are co-regulated, share common functionality, location or process, or directly bind one another; and directed edges may imply a step in a metabolic pathway, signal transduction cascade, stage of development, or a causal relationship [4]. These networks create the blackprint of the cellular system structure and provide design details of the cell.

Research in computational systems biology revolves around inferring or reverse engineering GRNs based on gene expression levels [5]. A basic assumption within the field is that the observed data, which are the changes in mRNA expression profiles, can explain transcriptional regulation. By inferring the underlying gene regulatory network from these large-scale experiments, ultimately the molecular role can be understood. The expression levels are the output of specific gene regulatory networks and therefore many algorithms have been studied to reverse engineer the GRNs most likely to produce observed expression data. Numerous issues arise from modeling GRNs from experimental data and therefore no one modeling technique outperforms all others. There are a vast number of genes and potential relationships; the experimentation to measure expression levels often result in noisy data; and there may be unobserved factors affecting the activity of genes that are not represented in the experiments conducted [1,6]. Once the networks are modeled, the topologies are scored to determine which are most consistent with the data. However, even the simplest GRNs are complex systems and difficult to infer.

Active research in reverse engineering of GRNs is conducted by testing different mathematical methods on computer generated networks where the true network is known. This allows for both validation and analysis of various inference algorithms. There are a few notable models commonly used for inferring GRNs: boolean networks [7], differential equations and linearization [8], regression methods [9], Gaussian models [10], conditional correlation analysis [11], and static and dynamic Bayesian networks [12,13]. Each provides advantages and disadvantages when inferring topologies [14]. Ultimately, the goal is to reverse engineer networks with confidence that the output of the statistical model is representative of the biological system.

1.1. Bayesian networks

Bayesian network (BN) modeling is an approach that combines probability and graph theory which has been useful in recovering gene regulatory networks from data. They can be used to describe the relationship between variables in gene regulatory networks and are promising because they can capture multiple types of
relationships [15]. These networks describe the relationship at a qualitative level. At the qualitative level, the graphical model showcases the dependences between various genes, which are encoded in the structure of the directed graph. An example BN is depicted in Fig. 1 in which \( X = (X_1, \ldots, X_5) \) represents the genes and the edges represent the dependencies. Each term \( p(X_i|PA_i) \) is the probability for a variable conditioned on the set of parents \( PA_i \) of \( X_i \). Bayesian networks specify the joint distribution over all variables for the conditional distribution of the node given the parental relationship:

\[
p(X_1, \ldots, X_n) = \prod_{i=1}^{n} p(X_i|PA_i).
\]

Numerous experiments have been conducted on in silico data to compare Bayesian networks to other inference models. Margolin et al. [16] developed ARACNE (algorithm for the reconstruction of accurate cellular networks) as another algorithms for inferring GRNs. Their study compares ARACNE to BNs because BNs are so widely used in reverse engineering and as such, the authors claim they provide an ideal benchmark technique. BNs are among the most effective models because of their ability to account for the stochastic nature of gene expression profiles and the easy integration of prior knowledge [14,17].

BNs are directed acyclic graphs and therefore the topology produced by the predicted model will include directed edges. This allows for modeling gene expression levels which depend on the regulators (parents) in the network. Accurate directed predictions are more difficult than undirected predictions. Some algorithms (e.g. ARACNE) only produce undirected results since the undirected topologies still provide useful insight into the underlying structure.

Given observed expression data \( D \), a Bayesian network approach enables a quantitative assessment regarding the likelihood that directed graph \( G \) produces such data. The general Bayesian scoring metric from [1] is the posterior probability of graph \( G \) given \( D \):

\[
S(G : D) = \log p(G|D) = \log \frac{p(D|G)p(G)}{p(D)} = \log p(D|G) + \log p(G) + \text{constant}.
\]  

The goal is to maximize the Bayesian score in Eq. (1). This score provides the ability to evaluate the quality of candidate graphs when searching for the network topology. In particular, we employ the Bayesian Dirichlet Equivalence (BDe) score [18,19] to help learn the BN and evaluate candidate GRN. This score incorporates a likelihood equivalence assumption and also allows for the incorporation of prior knowledge [19]. If relationships between nodes are already known, this information can be incorporated into the model. The metric penalizes any graph not containing an edge provided in the prior network. Another advantage of the score is the penalization of overly complex structures and the preference of simpler models of equally good networks.

1.2. Search heuristics

Finding the network topology that maximizes the likelihood of expressing the observed data is NP-hard [20,21]. Since the search space is large and no efficient exact algorithms are known for this problem, heuristic search is commonly used. The goal of heuristic search is to find a near optimal solution quickly and efficiently.

One commonly used search heuristic is the greedy technique hill climbing [6,22,23]. Hill climbing is similar to gradient ascent except that no derivatives are necessary. Instead, this iterative approach evaluates solutions that are “near” the current solution and adopts a new solution if a better one is found in the local search space. Compared with other techniques, this greedy search is fast, computationally simple, and requires few tuning parameters. Hill climbing, however, is myopic and prone to premature convergence to poor local optima. Random restarts are incorporated to mitigate this issue and expand the search region by performing hundreds or thousands of hill climbing procedures from randomly generated initial locations in the search space [24]. Yu et al. [15] found hill climbing with random restarts superior to simulated annealing and genetic algorithms. Other local search methods have been applied to learning Bayesian networks outside of the scope of gene regulatory networks: genetic algorithms [25], tabu search [26], ant colony optimization [27], dynamic programming with Markov Chain Monte Carlo techniques [28,29] and swarm optimization [30].

While Bayesian networks continue to be widely studied in application of gene regulatory network inference, research on the search heuristics paired with GRN inference is relatively limited. To date no search algorithms which have been paired with GRN inference have been able to compete with both the speed and solution quality of hill climbing with random restarts. It is the focus of this study to introduce a search heuristic that outperforms this greedy approach without compromising computation time. In this investigation we propose the Divided Neighborhood Exploration Search (DNES) heuristic to be paired with the Bayesian network modeling framework and evaluate its performance in producing high quality GRN’s.

2. Methods

2.1. In silico data and inference

To accurately evaluate an inference method, the true network must be known. As such, in silico data must be used. In particular, a directed acyclic graph, \( G = (V, E) \) is constructed where \( V \) is the set of nodes, and \( E \) is the set of directed edges \( (i, j) \), with \( i, j \in V \). The constructed topology can then be used to generate data that simulates gene expression data using ordinary differential equations that relate the changes in gene transcript concentration to each gene and to external perturbations. An inference method is used to reverse engineer the original network from the data. In the present study, we compare the implementation of Bayesian networks with a known greedy technique versus the novel DNES algorithm. Since the true network \( G \) is known, the quality of the engineered network \( G' \) is assessed based on agreement between the topologies of \( G \) and \( G' \). Fig. 2 depicts the high-level process.

![Bayesian network example](attachment:image.png)

**Fig. 1.** Bayesian network example.
Two measures, positive predictive value and sensitivity, are commonly used as evaluation metrics to compare \( G \) and \( G' \) and thus evaluate the inference technique. Let \( TP, FP, TN, \) and \( FN \) denote "true positives", "false positives", "true negatives" and "false negatives", respectively. Table 1 describes the relationships of these characteristics with respect to evaluating GRN inferences. A true positive is when an edge exists in both \( G \) and \( G' \); a true negative is when an edge does not exist in either, a false positive is an error wherein the inferred topology includes an edge that is not in \( G \); and a false negative is when the inferred topology excludes an edge which does exist in \( G \). Positive predicted value (PPV) is defined as

\[
PPV = \frac{TP}{TP + FP}
\]

and sensitivity (SE) is given by

\[
SE = \frac{TP}{TP + FN}
\]

Bayesian networks can be inferred from both large steady-state expression data and time series data \([12,31]\). Steady-state expression data represents data collected from perturbation experiments in which one or multiple genes are perturbed and the expression levels for all other genes are measured. Static (as opposed to dynamic) Bayesian networks are appropriate for steady-state perturbation experiments. We employ static BNs as the inference tool in this study.

While expression data from microarray experiments are continuous, for many genes, transcription occurs in only a small number of states. For example, it will occur when the expression level is low or high and does not occur between these states. Furthermore, when expression levels are measured significant noise is introduced and the data becomes more robust with respect to error when discretized \([32]\). Discretization of data is often employed prior to BN analysis \([e.g., 15,33,34]\). The data for this investigation are discretized using a three-category interval method. Two-category discretization creates high precision while networks reconstructed using four-category have had lower recall \([15]\). Three-category balances the loss of information due to overly coarse discretization and the difficulty of recovering links to nodes with multiple parents due to finer discretization.

The goal is to evaluate the effects of changing the search heuristic component of the inference algorithm to find the best solution within the search space. Bayesian networks were chosen as the inference component because of their popularity in evaluating GRNs. The novel search heuristic, DNES is implemented and compared with hill climbing with random restarts.

### 2.2. Divided neighborhood exploration search design

DNES is a form of variable neighborhood search (VNS) \([35]\), a technique which systematically modifies neighborhoods during a search to emerge from a local optimum. It incorporates many neighborhood structures whereas a local search typically only searches through one structure. VNS has been applied successfully to a wide variety of problems \([36–38]\). The DNES method uses VNS as the global search and variable neighborhood descent (VND) implemented as the local search component. Combining VNS with VND has led to successful applications \([39]\). VNS and VND will now be briefly described.

VNS depends on a finite number of pre-defined neighborhood structures. Often, VNS implementations use neighborhood definitions which are in some way nested structures. That is, neighborhood structure \( k \) is a subset of structure \( k + 1 \), or the scope of the neighborhood is increasing \( e.g., \) the 3-opt neighborhood used in the Traveling Salesman Problem is larger than the 2-opt neighborhood). However, there has been success using non-nested neighborhoods \([38]\).

Assume \( k = 1, \ldots, k_{\text{max}} \) neighborhood structures have been defined. Let \( x \) denote a solution in the search space. In the case of reverse engineering gene regulatory networks, a solution is an instance of a network topology \( i.e., \) a set of nodes and edges. In general, let \( N_k(x) \) denote the set of solutions in the \( k \)-th neighborhood of \( x \). The global search component of the general VNS approach randomly chooses and evaluates a different solution \( x' \in N_k(x) \). A local search is then engaged to attempt to improve on the quality of \( x \) by searching the neighborhood of \( x' \). If a better solution, \( x'' \), is found, the global search moves from solution \( x \) to \( x'' \). Otherwise, the procedure repeats using \( N_{k+1}(x) \) as the set of neighbors. The basic VNS algorithm is detailed in Fig. 3.

DNES uses VND as the local search mechanism. VND shares similarities with VNS except notably the stochastic element is eliminated. During the VND local search of \( x' \), every element of \( N_k(x') \) is evaluated. If the best value \( x'' \in N_k(x') \) is better than \( x' \), then the VND moves to the new solution \( x'' \) and continues the local search within \( N_k(x'') \). Otherwise, if no improvement on \( x' \) is found within \( N_k(x') \), \( k \) is incremented, and the local search continues in \( N_{k+1}(x') \). Once all \( k_{\text{max}} \) neighborhoods have been examined, the best solution discovered is returned to the global search procedure. The VND local search algorithm is depicted in Fig. 4.

The most important design decision for variable neighborhood search relates to the neighborhood structure definitions. A typical hill climbing approach to GRN applications defines the search neighborhood of graph \( G \) as every graph obtainable by a single addition, deletion, or reversal of an edge. DNES divides this larger neighborhood into three smaller neighborhood structures based on the allowable moves: add, reverse, and delete. The add neighbor-
hood of $G$ is defined as every acyclic graph obtainable by a single edge addition to $G$; the reverse neighborhood is defined as every acyclic graph obtainable by a directional reversal of an edge already existing in $G$; and, the delete neighborhood contains all graphs obtainable by deleting a single edge of $G$. Respectively, denote the three neighborhoods as $N_{\text{add}}(G)$, $N_{\text{reverse}}(G)$, and $N_{\text{delete}}(G)$ of graph $G$. Fig. 5 depicts a simple example. For a graph with $n$ nodes and $m$ edges and ignoring acyclic requirements, the add, reverse, and delete neighborhoods contain $n(n/2 - m, m)$, and $m$ neighbors, respectively. A move that causes a cycle is not feasible and therefore is excluded from evaluation.

The DNES implementation has $k_{\text{max}} = 3$ neighborhood structures. The design is meant to evaluate the effectiveness of dividing the exploration of the space into three non-overlapping move-based rules. For a given solution $G$ and the associated 1st and $j$th neighborhoods, $N_i(G) \cap N_j(G) = \emptyset$ for all $i \neq j$. This DNES variation of VNS with respect to partitioning of search moves is similar conceptually to the variant proposed in [38]. In their work they considered 19 different neighborhoods, each inherently mutually exclusive. They also note that the ordering of the neighborhood structures is important for non-nested designs since VNS is biased towards the first neighborhood structure.

In this study we evaluate two orderings: add, reverse, delete and secondly, delete, reverse, add. In the former, the search is biased towards adding more edges. In the latter, the search is biased towards deletions. Using the add neighborhood as the initial structure may produce denser networks than the alternative. Empirical analysis of these two permutations will provide more insight regarding the order-based performance.

Bayesian Network Inference with Java Objects (Banjo) [15] is the open-source software framework that supports the implementation of DNES. Specifically, Banjo provides the mechanism for computing the BDe score for a given network topology. Banjo has been evaluated and compared with other inference tools and was found to perform well with respect to positive predicted value, but less so with respect to sensitivity [5,40]. DNES is implemented through a variety of changes to the Banjo search framework, however the scoring mechanism is left intact.

### 2.3. Experimental design

To evaluate the effectiveness of the novel search mechanism, a total of 200 sets of gene networks are considered. This data comes from two primary sources. The first of which are in silico networks and expression data developed and successfully evaluated in Bansal et al. [5]. Each of these directed networks is composed of either 10 or 100 genes with an average in-degree per gene of 2 or 10, respectively. There are twenty 10 gene networks and twenty 100 gene networks. The corresponding expression data is generated using linear ODEs that relate the changes in gene transcript concentration to each gene and to external perturbations to simulate knockdown experiments. The simulated data includes "white
noise" as an error term (with zero mean and standard deviation equal to one tenth the absolute value of the expression level) to simulate the noise that occurs during measurement.

The second set of in silico data are generated using GeneNetWeaver [41]. GeneNetWeaver (GNW) is an open-source application for generating networks by extracting subnetworks from known biological interaction networks. Such an extraction process allows researchers to use gold standard networks which are reflective of more realistic gene network topologies [42]. The tool allows users to define various parameters of the in silico data including the number of nodes in the subnetwork, type of simulated experiment (e.g., knockout or knockdown), and how much noise to introduce into the data. Synthetic expression data from the GNW knockout experimentation reflect steady-state levels of single-gene knockouts – the transcription rates of each gene are independently set to zero one at a time. A similar method is used for simulated knockdown expression data, except the transcription rates are reduced by half instead of set to zero. Search heuristics may per-

![Fig. 5. Add, reverse, and delete neighborhoods of graph G.](image)

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Search methods.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notation</td>
<td>Approach</td>
</tr>
<tr>
<td>LG500</td>
<td>Local greedy</td>
</tr>
<tr>
<td>LG1000</td>
<td>Local greedy</td>
</tr>
<tr>
<td>RG500</td>
<td>Random greedy</td>
</tr>
<tr>
<td>RG1000</td>
<td>Random greedy</td>
</tr>
<tr>
<td>A/R/D</td>
<td>DNES Add/Reverse/Delete</td>
</tr>
<tr>
<td>D/R/A</td>
<td>DNES Delete/Reverse/Add</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 3</th>
<th>Summary of search algorithm metrics on benchmark data from [5].</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genes</td>
<td>Technique</td>
</tr>
<tr>
<td>---------</td>
<td>-----------</td>
</tr>
<tr>
<td>10</td>
<td>LG500</td>
</tr>
<tr>
<td></td>
<td>LG1000</td>
</tr>
<tr>
<td></td>
<td>RG500</td>
</tr>
<tr>
<td></td>
<td>RG1000</td>
</tr>
<tr>
<td></td>
<td>A/R/D</td>
</tr>
<tr>
<td></td>
<td>D/R/A</td>
</tr>
<tr>
<td>100</td>
<td>LG500</td>
</tr>
<tr>
<td></td>
<td>LG1000</td>
</tr>
<tr>
<td></td>
<td>RG500</td>
</tr>
<tr>
<td></td>
<td>RG1000</td>
</tr>
<tr>
<td></td>
<td>A/R/D</td>
</tr>
<tr>
<td></td>
<td>D/R/A</td>
</tr>
</tbody>
</table>

### Table 4
Summary of search algorithm inferred edges on benchmark data from [5].

<table>
<thead>
<tr>
<th>Genes Technique</th>
<th>Undirected Genes</th>
<th>Directed Genes</th>
<th>Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TP</td>
<td>FP</td>
<td>FN</td>
</tr>
<tr>
<td>10 LG500</td>
<td>4.0 (2.15)</td>
<td>2.0 (2.53)</td>
<td>13.5 (3.03)</td>
</tr>
<tr>
<td>LG1000</td>
<td>4.1 (2.16)</td>
<td>2.0 (2.48)</td>
<td>13.5 (3.05)</td>
</tr>
<tr>
<td>RG500</td>
<td>4.0 (2.22)</td>
<td>1.8 (2.1)</td>
<td>13.5 (3.1)</td>
</tr>
<tr>
<td>RG1000</td>
<td>4.0 (2.22)</td>
<td>1.7 (1.95)</td>
<td>13.5 (3.1)</td>
</tr>
<tr>
<td>A/R/D</td>
<td>3.3 (1.95)</td>
<td>0.9 (1.27)</td>
<td>14.4 (3.03)</td>
</tr>
<tr>
<td>D/R/A</td>
<td>3.4 (2.06)</td>
<td>1.0 (1.43)</td>
<td>14.4 (3.05)</td>
</tr>
</tbody>
</table>

### Table 5
Summary of search algorithm metrics on GNW data.

<table>
<thead>
<tr>
<th>Type</th>
<th>Genes Technique</th>
<th>PPV</th>
<th>SEU</th>
<th>PPVD</th>
<th>SEU</th>
<th>BDe</th>
<th>Time</th>
<th>Networks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knockout 100</td>
<td>LG500</td>
<td>0.053</td>
<td>0.032</td>
<td>0.025</td>
<td>0.016</td>
<td>-1621.1</td>
<td>58.66</td>
<td>5,054,899</td>
</tr>
<tr>
<td></td>
<td>LG1000</td>
<td>0.053</td>
<td>0.033</td>
<td>0.026</td>
<td>0.018</td>
<td>-1612.5</td>
<td>115.63</td>
<td>10,109,871</td>
</tr>
<tr>
<td></td>
<td>RG500</td>
<td>0.186</td>
<td>0.104</td>
<td>0.122</td>
<td>0.067</td>
<td>-1114.2</td>
<td>15.11</td>
<td>2,458,185</td>
</tr>
<tr>
<td></td>
<td>RG1000</td>
<td>0.248</td>
<td>0.52</td>
<td>0.962</td>
<td>0.149</td>
<td>14.8</td>
<td>974.99</td>
<td>29.7</td>
</tr>
<tr>
<td>A/R/D</td>
<td>49.9</td>
<td>0.102</td>
<td>0.043</td>
<td>0.057</td>
<td>0.025</td>
<td>-2889.2</td>
<td>51.46</td>
<td>2,494,923</td>
</tr>
<tr>
<td>D/R/A</td>
<td>49.9</td>
<td>0.098</td>
<td>0.043</td>
<td>0.057</td>
<td>0.025</td>
<td>-2880.9</td>
<td>51.46</td>
<td>2,494,923</td>
</tr>
<tr>
<td>Knockdown 100</td>
<td>LG500</td>
<td>0.060</td>
<td>0.028</td>
<td>0.027</td>
<td>0.013</td>
<td>-1535</td>
<td>58.5</td>
<td>5,054,899</td>
</tr>
<tr>
<td></td>
<td>LG1000</td>
<td>0.058</td>
<td>0.029</td>
<td>0.026</td>
<td>0.014</td>
<td>-1529</td>
<td>116.2</td>
<td>10,109,871</td>
</tr>
<tr>
<td></td>
<td>RG500</td>
<td>0.215</td>
<td>0.093</td>
<td>0.142</td>
<td>0.063</td>
<td>-1087</td>
<td>13.3</td>
<td>2,450,306</td>
</tr>
<tr>
<td></td>
<td>RG1000</td>
<td>0.208</td>
<td>0.092</td>
<td>0.134</td>
<td>0.087</td>
<td>28.5</td>
<td>5,028,542</td>
<td></td>
</tr>
<tr>
<td>A/R/D</td>
<td>49.9</td>
<td>0.102</td>
<td>0.043</td>
<td>0.057</td>
<td>0.025</td>
<td>-2889.2</td>
<td>51.46</td>
<td>2,494,923</td>
</tr>
<tr>
<td>D/R/A</td>
<td>49.9</td>
<td>0.098</td>
<td>0.043</td>
<td>0.057</td>
<td>0.025</td>
<td>-2880.9</td>
<td>51.46</td>
<td>2,494,923</td>
</tr>
</tbody>
</table>

form differently with knockout expression data compared with knockdown expression data since the latter is less informative [41].

Currently, there are two source networks available: E. coli with 1,565 nodes and 3,158 edges and the yeast S. cerevisiae having 4,441 nodes and 12,873 edges. Our experimentation is conducted on twenty instances each of knockdown and knockout simulated expression data sets used.

DNES is evaluated against the greedy search approach hill climbing with random restarts. Banjo has two different hill climbing implementations which we refer to as random greedy and local greedy. The first randomly examines a single change in the network, compares the BDe score to the previous topology, and potentially moves to the proposed network based on this comparison. The second generates a list of all feasible moves within a solution neighborhood and evaluates each network before deciding whether or not to move. In both cases the user defines the number of random restarts. The parameters 500 and 1000 have been set for the number of restarts used with both variations of the greedy search. DNES does not utilize random restarts because of the systematic approach to avoid local optimum. Table 2 outlines the search methods, the notation used, and the settings evaluated.

Each search approach requires an initial topology. We randomly generate initial acyclic networks. The initial random network is the same for each search approach for a given test. That is, when inferring network 1, all six searchers begin with the same initial structure. To infer network 2, a different random topology is used to initialize all six searchers.

Several performance metrics are computed and compared for each search technique including: PPV, SE, BDe score, and computation time. Note that the BDe score is the basis for network scoring and thus the evaluation function that drives all search techniques. If the DNES approach is successful, then it should find higher BDe scores than the alternative techniques. PPV and SE are reported in terms of both directed and undirected network evaluations. PPVD and SEU denote the PPV and SE with respect to an undirected topology, whereas PPVD and SED correspond to a directed graph. That is, suppose edge \((i, j)\) is in Network 1, if a searcher infers edge \((j, i)\), this would be considered a true positive when computing PPVD and SEU, but a false positive in the PPVD and SED calculation.

### 3. Results and discussion

The four hill climbing variations in the experimentation are random greedy with 500 or 1000 restarts (denoted RG500 and RG1000, respectively) and local greedy with 500 or 1000 restarts (denoted LG500 and LG1000, respectively). The two orders for the DNES search are denoted A/R/D and D/R/A. Using this notation, the aver-

<table>
<thead>
<tr>
<th>Type</th>
<th>Genes</th>
<th>Technique</th>
<th>Undirected</th>
<th>Directed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>TP</td>
<td>FP</td>
</tr>
<tr>
<td>Knockout</td>
<td>100</td>
<td>LG500</td>
<td>7.2</td>
<td>132.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(4.75)</td>
<td>(52.4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LG1000</td>
<td>7.3</td>
<td>133.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(4.91)</td>
<td>(53.1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RG500</td>
<td>24.3</td>
<td>105.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(11.9)</td>
<td>(24.9)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RG1000</td>
<td>23.4</td>
<td>106.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(12.11)</td>
<td>(25.1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A/R/D</td>
<td>57.5</td>
<td>117.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(20.33)</td>
<td>(57.3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D/R/A</td>
<td>57.9</td>
<td>116.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(21.24)</td>
<td>(56.8)</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>RG500</td>
<td>22.2</td>
<td>199.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(8.49)</td>
<td>(33.6)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RG1000</td>
<td>21.7</td>
<td>201.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(7.95)</td>
<td>(32)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A/R/D</td>
<td>110.9</td>
<td>262.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(38.28)</td>
<td>(95.8)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D/R/A</td>
<td>110.5</td>
<td>261.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(37.97)</td>
<td>(96.6)</td>
</tr>
<tr>
<td>Knockdown</td>
<td>100</td>
<td>LG500</td>
<td>6.5</td>
<td>112.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(4.12)</td>
<td>(43.1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LG1000</td>
<td>6.6</td>
<td>118.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(4.11)</td>
<td>(46.6)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RG500</td>
<td>21.2</td>
<td>82.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(9.4)</td>
<td>(19.2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RG1000</td>
<td>20.6</td>
<td>83.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(9.14)</td>
<td>(19)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A/R/D</td>
<td>52.9</td>
<td>82.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(18.09)</td>
<td>(48)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D/R/A</td>
<td>53.0</td>
<td>81.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(18.16)</td>
<td>(47.2)</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>RG500</td>
<td>17.7</td>
<td>153.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(4.39)</td>
<td>(30.6)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RG1000</td>
<td>17.4</td>
<td>153.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(3.55)</td>
<td>(30.1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A/R/D</td>
<td>94.0</td>
<td>208.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(32.47)</td>
<td>(72.9)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D/R/A</td>
<td>94.7</td>
<td>207.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(32.58)</td>
<td>(72.9)</td>
</tr>
</tbody>
</table>
age PPV_U, SE_U, PPV_D, SE_D, BDe, computational time in seconds, and total number of networks searched are reported in Table 3 for the benchmark data from [5]. Table 4 also reports information specific to the average number for TP, FP, and FN for the undirected and directed evaluations as well as the total number of edges in the inferred topology. In both Tables 3 and 4, the standard deviations of the various metrics are listed in parentheses below the associated mean values. Recall that each summary statistic is associated with the results of experimentation on 20 networks. Tables 5 and 6 report similar results for the tests on the GNW data which is also broken out by type of experimentation simulated: knockout or knockdown. Again, for Tables 5 and 6, the standard deviations are listed in parentheses below the mean values. Since there are 20 subnetworks derived from \textit{E. coli} and 20 derived from \textit{S. cerevisiae} for each setting (100 or 200 nodes), the means and standard deviations correspond to search results performed on 40 networks. Note that in these tables, results are not reported using the local greedy approaches for the 200 GNW gene networks. After

Fig. 6. Estimated densities of BDe scores for 100 gene benchmarks from [5].

Fig. 7. Estimated densities of BDe scores for 100 gene GNW networks.
initial analysis, the LG500 and LG1000 run times were observed to exceed 20 min for every test and the performance metrics were consistently poor. Whether this is an issue related to memory usage or simply poor performance of the local greedy method we cannot say. Intuitively, on larger networks, the performance of local greedy approaches will suffer since at each iteration the local neighborhoods must be evaluated exhaustively.

The 10 gene test results in Table 3 demonstrate that the DNES techniques search far fewer candidate networks (three orders of magnitude less) than the greedy heuristic methods for these relatively simple problems. However, after examining only about 1,000 networks, the PPV values and BDe scores are comparable to those found by the greedy methods which searched through more than 1.5 M possible topologies. A similar pattern occurs in the more interesting and larger 100 gene networks. The DNES techniques search through a fraction of the number of networks (in a fraction of the time) to obtain comparable or better scores in all the reported statistics.

The 100 gene experiments show that the random greedy and DNES methods produce results with high PPV values for the undirected graphs. As expected, the PPV\textsubscript{U} values are notably less than the PPV\textsubscript{D} values. The local greedy techniques perform poorly with respect to both PPV\textsubscript{U} and PPV\textsubscript{D}. This is possibly due to the emphasis on local searching over wider exploration during even the earliest iterations: it is unlikely that the initial random solutions are near “good” solutions. The overall lower sensitivities observed for all techniques is consistent with literature when no prior knowledge is incorporated [5,45].

Tables 3 and 4 reveal that the 500 additional random restarts used in the greedy approaches has limited impact. This is especially true for the local greedy searches in which the LG500 and LG1000 results have nearly identical values for every performance

Table 7
Paired performance differences on benchmark data from [5].

<table>
<thead>
<tr>
<th>Genes</th>
<th>Techniques</th>
<th>Undirected</th>
<th>Directed</th>
<th>Edges</th>
<th>BDe</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>PPV\textsubscript{U}</td>
<td>SE\textsubscript{U}</td>
<td>TP</td>
<td>PPV\textsubscript{D}</td>
<td>SE\textsubscript{D}</td>
</tr>
<tr>
<td>10</td>
<td>A/R/D – RG500</td>
<td>0.06</td>
<td>(0.11)</td>
<td>(&lt;0.01)</td>
<td>0.04</td>
<td>(0.05)</td>
</tr>
<tr>
<td></td>
<td>D/R/A – RG500</td>
<td>0.05</td>
<td>(0.25)</td>
<td>(0.01)</td>
<td>0.00</td>
<td>(0.94)</td>
</tr>
<tr>
<td></td>
<td>RG500 – LG500</td>
<td>0.00</td>
<td>(0.91)</td>
<td>(0.98)</td>
<td>1.00</td>
<td>(0.49)</td>
</tr>
<tr>
<td></td>
<td>A/R/D – D/R/A</td>
<td>0.01</td>
<td>(0.34)</td>
<td>0.00</td>
<td>–0.1</td>
<td>(0.37)</td>
</tr>
<tr>
<td>100</td>
<td>A/R/D – RG500</td>
<td>0.00</td>
<td>(0.20)</td>
<td>(&lt;0.01)</td>
<td>0.00</td>
<td>(0.99)</td>
</tr>
<tr>
<td></td>
<td>D/R/A – RG500</td>
<td>0.00</td>
<td>(0.15)</td>
<td>(&lt;0.01)</td>
<td>0.00</td>
<td>(0.70)</td>
</tr>
<tr>
<td></td>
<td>RG500 – LG500</td>
<td>0.00</td>
<td>(&lt;0.01)</td>
<td>(0.12)</td>
<td>0.12</td>
<td>(&lt;0.01)</td>
</tr>
<tr>
<td></td>
<td>A/R/D – D/R/A</td>
<td>0.00</td>
<td>(0.32)</td>
<td>0.00</td>
<td>0.00</td>
<td>(0.29)</td>
</tr>
</tbody>
</table>
metrics except time and the number of networks searched. Table 5 shows that for both knockout and knockdown experiments, for both 100 and 200 gene networks, and for both directed and undirected metrics, the DNES methods obtain the largest means values of PPV and SE. The inferred topologies have both higher average true positives and generally have more edges as seen in Table 6. Both neighborhood orderings of the DNES techniques produce comparable performance metrics and network sizes.

A key element of interest is the shift which occurs regarding the total number of networks searched and computation time for the DNES methods. For the 200 gene networks, DNES evaluates for more networks than the random greedy techniques. We expect this to be due to one or more factors which are discussed in the conclusion.

The mean BDe values associated with DNES in Table 5 are consistently the least negative across the experiments. Figs. 6–8 depict the density of the BDe scores for a subset of the search techniques (LG500, RG500, and D/R/A) for 100 gene data from [5], the 100 gene GNW data, and the 200 gene GNW data, respectively. Fig. 6 shows that for the simulated data from [5] there is overlap of BDe scores achieved by the D/R/A method and the two greedy approaches. The separation among all three distributions improves when analyzing the more realistic GNW data. Fig. 7 shows that D/R/A is likely to be statistically superior to either of the greedy approaches. The separation among all three distributions improves when analyzing the more realistic GNW data. For the larger 200 gene networks, the densities in Fig. 8 are clearly separated and D/R/A produces the better BDe values. Statistical comparisons of the D/R/A performance with both the RG500 and the A/R/D performance data follow.

Four approaches are selected for statistical analysis: RG500, LG500, A/R/D, and D/R/A. The RG1000 and LG1000 approaches are excluded since the additional restarts did little if anything to improve the performance metrics. A paired t-test is used since each experiment is independent and the data follow.

Paired performance differences on GNW data.

Table 8 shows the DNES methods produce statistically better PPVU, PPVD, SEU, and SED values for all the GNW experimentation. The DNES inferred topologies are also larger (contain more edges) and have more directed and undirected true positives. With regards to comparing performances of the two neighborhood orderings for DNES, Tables 7 and 8 shows there is no significant difference in any metrics between the orderings.

4. Conclusion

In this study we introduce a search heuristic, DNES, to be used as the metaheuristic search component of an inference algorithm to improve the quality of reverse engineered GRNs without compromising the fast computation times of the standard greedy search heuristics.
search technique. The empirical analysis, which leverages a Baye-
sian network scoring algorithm, demonstrates that stochastic
greedy approaches (i.e., RG500 and RG1000) significantly outper-
form the deterministic greedy approaches (LG500 and LG1000)
which exhaust the local search space. High quality solutions are
unlikely to be close in proximity to the initial solution and there-
fore movement through the search space is a vital component.

DNES combines elements of both stochastic and deterministic
greedy techniques. In the first phase it randomly chooses a local
point for exploration and then exhaustively searches the local
space of this point to improve on solution quality. This combina-
tion of exploration and exploitation is demonstrated as an effective
method for discovering better topologies. Additionally, DNES elim-
nates the need for random restarts to find good solutions. The sys-
tematic switching of neighborhood structures allows the search
procedure to escape local optima. DNES finds better quality solu-
tions than the four hill climbing with random restarts implementa-
tions analyzed. The current implementation of DNES as a
modification to Banjo is available at http://oklahomaanalyt-
ics.com/software-research-data.

The empirical analysis reveals that an increase in the number of
networks explored does not guarantee better results. The LG500
implementation evaluates an average of over 5 million networks
for the 100 gene experiments yet under performs RG500 which
evaluates about half as many. For the 100 gene GNW experiments, DNES
evaluates a fraction of these quantities and produces better
inferred topologies. The simultaneous reduction in the number of
network evaluations and improvement in solution quality demon-
strates that the search space can be explored more effectively with
an intelligent partitioning of the search moves. Dividing the single
hill climbing neighborhood into three separate structures and using
a combination of systematic and stochastic evaluations results in
better BDe values, comparable positive predictive values, and
improved sensitivity.

The issue regarding the relatively large DNES searches with the
GNW 200 gene may be due to one or more factors. The first is that
Banjo is over restricting the random greedy search for the RG500
and RG1000 methods and they are terminating too soon. However,
since there is little to no improvement when the restarts are incre-
mented from 500 to 1000 restarts, it simply may be that an unre-
stricted searcher will not be productive. The second potential
factor is that as the networks increase in size, the DNES neighbor-
hoods do as well. The local search component of DNES is an
exhaustive search. If the local search neighborhood is defined to
be too broad, then the number of topologies examined may
increase dramatically. The local greedy techniques demonstrate
that exhaustive, deterministic searching can be problematic. How-
ever, the unrestricted DNES algorithm appears to be productive so
far. The increased quantity of networks searched allowed DNES to
find statistical superior topologies.

While other advanced metaheuristics may outperform the
greedy techniques in terms of quality, they are not used in practice
because of the computational burden. Yu et al. [15] compared sim-
ulated annealing, a genetic algorithm, and hill climbing with 100
random restarts and concluded the greedy approach was the most
practical because of the comparable output and the computation
time was a fraction of the other techniques. DNES, however, consis-
tently terminates faster than the greedy approaches on 100 gene
networks evaluated and was only seconds slower than RG500 for
the 200 gene networks. It seems that the simple idea of dividing
the typical neighborhood definition into three distinct neighbor-
hoods and using a combination of stochastic and deterministic
searches perform very well. The algorithm is simply to implement
and requires no tuning parameters (unlike genetic algorithms). DNES
does not even require identifying the appropriate number of
restarts. The method we propose is simple, produces better qual-
ity solutions than hill climbing with restarts, and has practical and
promising search times.

References

[1] V. Filkov, Identifying gene regulatory networks from gene expression data, in:
Handbook of Computational Molecular Biology, Chapman&Hall, CRC Press,
2005.
analysis algorithms, Bioinformatics 19 (Suppl. 2) (2003) i112–i120.
perturbation: inference and experimental design, Pacific Symposium on
[8] P. Chen, H. Ha, G. Church, Modeling gene expression with differential
equations, Pacific Symposium on Biocomputing, vol. 4, World Scientific,
and identifying compound mode of action via expression profiling, Science 301
for biological systems, in: Research in Computational Molecular Biology,
[12] N. Friedman, M. Linial, I. Nachman, D. Pe'er, Using bayesian networks to
Gene networks inference using dynamic bayesian networks, Bioinformatics
inference for generating causal networks from observational biological data,
Califano, ARACNE: an algorithm for the reconstruction of gene regulatory
networks in a mammalian cellular context, BMC Bioinform. 7 (Suppl. 1) (2006)
57.
[17] B. Ristevski, Overview of computational approaches for inference of microcra-
mediated and gene regulatory networks, Advances in Computers, vol. 97,
[18] C. de Campos, D. Qi, Properties of bayesian dirichlet scores to learn bayesian
combination of knowledge and statistical data, Machine Learning 20 (3) (1995)
197–243.
inference algorithms to recover molecular genetic regulatory networks, in: 3rd
International Conference on Systems Biology, 2002.
[23] Q. Zhang, Y. Cao, Y. Li, Y. Zhu, S. Sun, D. Guo, A sub-space greedy search method
763–770.
efficient methods based on progressive restriction of the neighborhood, Data
of bayesian networks by genetic algorithms: a performance analysis of control
[26] S. Acid, L. de Campos, J. Castellano, Learning bayesian network classifiers:
searching in a space of partially directed acyclic graphs, Machine Learning 59
[27] L. De Campos, J. Fernandez-Luna, J. Gámez, J. Puerta, Ant colony optimization
[28] D. Eaton, K. Murphy, Bayesian structure learning using dynamic programming
and mcmc, in: 23rd Conference on Uncertainty in Artificial Intelligence (UAI
2007), 2012.
[29] M. Grzegorczyk, D. Husmeier, Improving the structure mcmc sampler for
bayesian networks by introducing a new edge reversal move, Machine Learning