Journal of Biomedical Informatics 65 (2017) 120–131

Contents lists available at ScienceDirect

Journal of Biomedical Informatics

journal homepage: www.elsevier.com/locate/yjbin

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

ARTICLE INFO

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

ABSTRACT

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

* Corresponding author.
E-mail addresses: cnicholson@ou.edu (C. Nicholson), leslie.goodwin@ou.edu (L. Goodwin), coreyc@smu.edu (C. Clark).

http://dx.doi.org/10.1016/j.jbi.2016.11.010
1532-0464/

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 $\mathbf{X} = (X_1, \ldots, X_5)$ represents the genes and the edges represent the dependencies. Each term $p(X_i|\mathrm{PA}_i)$ 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:

$$\begin{aligned} S(G:D) &= \log P(G|D) = \log \frac{p(D|G)p(G)}{p(D)} \\ &= \log p(D|G) + \log p(G) + \text{constant}. \end{aligned} \quad (1)$$

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.

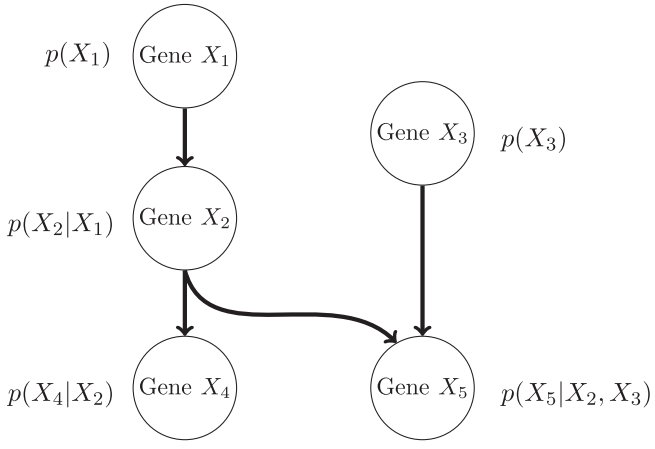

Fig. 1. Bayesian network example.

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.

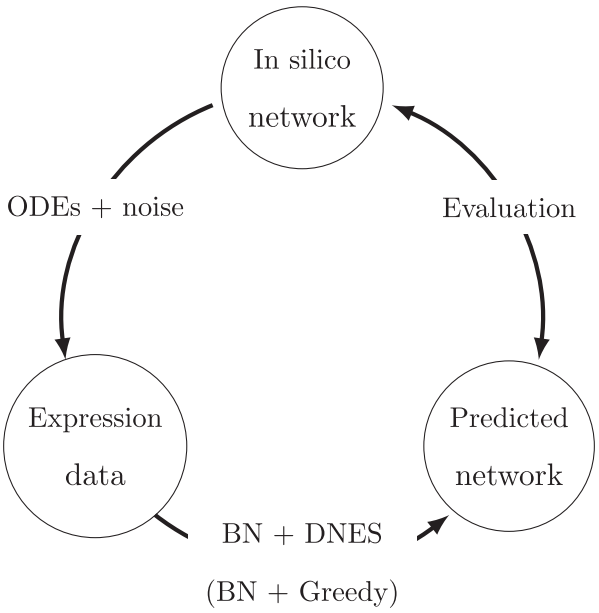

Fig. 2. Overview of *in silico* data generation and search heuristic evaluation process.

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

$$\text{PPV} = \frac{TP}{TP + FP}$$

and sensitivity (SE) is given by

$$\text{SE} = \frac{TP}{TP + FN}$$

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 imprecision 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.

Table 1
TP, FN, FP, TN definition with respect to edge (i,j) as an element of the true graph G and the predicted graph G'.

	$(i,j) \in G$	$(i,j) \notin G$
$(i,j) \in G'$	TP	FP
$(i,j) \notin G'$	FN	TN

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 $k = 1, \ldots, k_{max}$ 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 $\mathcal{N}_k(x)$ denote the set of solutions in the kth neighborhood of x. The global search component of the general VNS approach randomly chooses and evaluates a different solution $x' \in \mathcal{N}_k(x)$. 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 $\mathcal{N}_{k+1}(x)$ 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 $\mathcal{N}_k(x')$ is evaluated. If the best value $x'' \in \mathcal{N}_k(x')$ is better than x', then the VND moves to the new solution x'' and continues the local search within $\mathcal{N}_k(x'')$. Otherwise, if no improvement on x' is found within $\mathcal{N}_k(x'), k$ is incremented, and the local search continues in $\mathcal{N}_{k+1}(x')$. Once all k_{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-*

Data: inital solution x, $k_{\max}$ neighborhood structures, function f to evaluate solutions

Result: a potentially improved x

begin

$k \leftarrow 1$

while $k \leq k_{max}$ **do**

$x' \leftarrow$ random element of $\mathcal{N}_k(x)$

$x'' \leftarrow LocalSearch(x')$

if $f(x'') > f(x)$ **then**

$x \leftarrow x''$

else

$k \leftarrow k + 1$

Fig. 3. Variable neighborhood search algorithm.

Data: inital solution x', $k_{\max}$ neighborhood structures, function f to evaluate solutions

Result: a potentially improved x'

begin

$k \leftarrow 1$

while $k \leq k_{max}$ **do**

Find the best neighbor $x'' \in \mathcal{N}_k(x')$

if $f(x'') > f(x')$ **then**

$x' \leftarrow x''$

else

$k \leftarrow k + 1$

Fig. 4. Variable neighborhood descent algorithm used as *LocalSearch* of VNS.

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 $\mathcal{N}_{\text{add}}(G), \mathcal{N}_{\text{reverse}}(G)$, and $\mathcal{N}_{\text{delete}}(G)$ 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 $\frac{n(n-1)}{2} - m, m$, and m neighbors, respectively. A move that causes a cycle is not feasible and therefore is excluded from evaluation.

The DNES implementation has $k_{max} = 3$ 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 ith and jth neighborhoods, $\mathcal{N}_i(G) \cap \mathcal{N}_j(G) = \varnothing$ 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

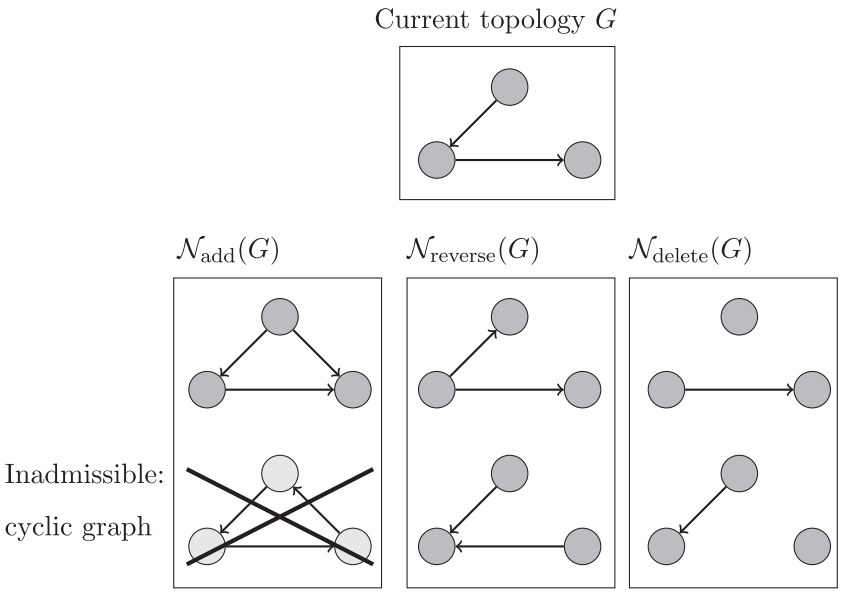

Fig. 5. Add, reverse, and delete neighborhoods of graph G.

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.

Table 2
Search methods.

Notation	Approach	Number of restarts
LG500	Local greedy	500
LG1000	Local greedy	1000
RG500	Random greedy	500
RG1000	Random greedy	1000
A/R/D	DNES Add/Reverse/Delete	0
D/R/A	DNES Delete/Reverse/Add	0

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-

Table 3
Summary of search algorithm metrics on benchmark data from [5].

Genes	Technique	PPV_U	SE_U	PPV_D	SE_D	BDe	Time	Networks
10	LG500	0.787	0.234	0.461	0.138	−65	1.28	1,573,674
		(0.23)	(0.13)	(0.24)	(0.09)	(5.9)	(0.26)	(30,062)
	LG1000	0.793	0.237	0.461	0.138	−65	2.23	3,146,977
		(0.22)	(0.13)	(0.24)	(0.09)	(5.9)	(0.47)	(59,445)
	RG500	0.79	0.234	0.419	0.132	−65	0.83	1,500,495
		(0.20)	(0.14)	(0.20)	(0.09)	(5.9)	(0.13)	(1119)
	RG1000	0.796	0.234	0.425	0.132	−65	1.53	3,001,175
		(0.19)	(0.14)	(0.21)	(0.09)	(5.8)	(0.26)	(3313)
	A/R/D	0.856	0.193	0.383	0.105	−64	0.02	1092
		(0.192)	(0.12)	(0.31)	(0.10)	(7.1)	(0.01)	(455)
	D/R/A	0.843	0.196	0.413	0.111	−64	0.02	1168
		(0.22)	(0.12)	(0.31)	(0.10)	(6.8)	(0.01)	(352)
100	LG500	0.207	0.020	0.117	0.012	−1483	58.99	5,054,899
		(0.04)	(0.004)	(0.03)	(0.003)	(408.0)	(5.31)	(1329)
	LG1000	0.207	0.020	0.117	0.012	−1483	117.18	10,109,972
		(0.04)	(0.004)	(0.03)	(0.003)	(408.0)	(9.38)	(2716)
	RG500	0.855	0.024	0.524	0.015	−1266	8.28	2,243,000
		(0.12)	(0.01)	(0.10)	(0.01)	(346.7)	(2.06)	(262,332)
	RG1000	0.873	0.025	0.521	0.015	−1264	15.76	4,486,225
		(0.13)	(0.01)	(0.11)	(0.01)	(346.9)	(3.66)	(524,690)
	A/R/D	0.875	0.051	0.524	0.031	−1167	1.74	654,096
	(0.08)	(0.03)	(0.08)	(0.02)	(291.1)	(1.41)	(376,399)	
	D/R/A	0.877	0.051	0.533	0.031	−1164	1.81	662,784
		(0.08)	(0.03)	(0.09)	(0.02)	(288.8)	(1.27)	(373,703)

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

Genes	Technique	Undirected			Directed			Edges
		TP	FP	FN	TP	FP	FN	
10	LG500	4.0	2.0	13.5	2.5	3.5	15.7	6.0
		(2.15)	(2.53)	(3.03)	(1.67)	(3.12)	(2.06)	(4.26)
	LG1000	4.1	2.0	13.5	2.5	3.5	15.7	6.0
		(2.16)	(2.48)	(3.05)	(1.67)	(3.12)	(2.06)	(4.26)
	RG500	4.0	1.8	13.5	2.4	3.4	15.8	5.8
		(2.22)	(2.1)	(3.1)	(1.6)	(2.87)	(2.02)	(4.01)
	RG1000	4.0	1.7	13.5	2.4	3.3	15.8	5.7
		(2.22)	(1.95)	(3.1)	(1.6)	(2.84)	(2.05)	(3.91)
	A/R/D	3.3	0.9	14.4	1.9	2.3	16.3	4.2
		(1.95)	(1.27)	(3.03)	(1.94)	(1.52)	(2.34)	(2.76)
	D/R/A	3.4	1.0	14.4	2.0	2.4	16.2	4.4
		(2.06)	(1.43)	(3.05)	(1.86)	(1.93)	(2.32)	(2.87)
100	LG500	20.3	77.2	967.9	11.4	86.1	978.3	97.5
		(4.12)	(5.25)	(4.79)	(2.72)	(6.21)	(3.61)	(6.54)
	LG1000	20.3	77.2	967.9	11.4	86.1	978.3	97.5
		(4.12)	(5.25)	(4.79)	(2.72)	(6.21)	(3.61)	(6.54)
	RG500	24.1	5.6	962.7	14.9	14.8	974.9	29.7
		(10.49)	(6.44)	(12.41)	(6.64)	(10.10)	(7.44)	(16.04)
	RG1000	24.8	5.2	962.4	14.9	15.1	974.8	30.0
		(10.66)	(6.64)	(12.50)	(6.67)	(10.23)	(7.51)	(16.18)
	A/R/D	49.9	10.2	935.2	30.9	29.2	958.9	60.0
		(27.51)	(11.38)	(31.43)	(18.32)	(20.77)	(18.71)	(38.06)
	D/R/A	49.9	10.0	935.2	30.9	29.0	958.8	59.9
		(27.47)	(11.29)	(31.40)	(18.33)	(20.75)	(18.65)	(37.91)

Table 5
Summary of search algorithm metrics on GNW data.

Type	Genes	Technique	PPV_U	SE_U	PPV_D	SE_D	BDe	Time	Networks
Knockout	100	LG500	0.053	0.032	0.025	0.016	−1621.1	58.66	5,054,899
			(0.03)	(0.02)	(0.02)	(0.01)	(247.5)	(2.59)	(1312)
		LG1000	0.053	0.033	0.026	0.018	−1612.5	115.63	10,109,871
			(0.03)	(0.02)	(0.02)	(0.01)	(248.7)	(2.26)	(2647)
		RG500	0.186	0.104	0.122	0.067	−1114.2	15.11	2,458,185
			(0.08)	(0.03)	(0.06)	(0.02)	(151.5)	(2.90)	(177,879)
		RG1000	0.180	0.102	0.116	0.065	−1129.0	32.15	5,044,725
			(0.09)	(0.03)	(0.06)	(0.02)	(193.3)	(15.04)	(894,366)
		A/R/D	0.354	0.274	0.288	0.219	−983.7	6.80	1,625,721
			(0.16)	(0.11)	(0.15)	(0.10)	(90.1)	(4.86)	(574,806)
		D/R/A	0.357	0.274	0.292	0.221	−981.9	6.87	1,632,420
			(0.16)	(0.11)	(0.15)	(0.1)	(87.7)	(5.01)	(592,159)
	200	RG500	0.101	0.043	0.059	0.025	−2889.2	51.46	2,494,923
			(0.04)	(0.01)	(0.03)	(0.01)	(445.4)	(9.36)	(30,596)
		RG1000	0.098	0.043	0.058	0.025	−2883.2	103.02	4,990,050
			(0.04)	(0.01)	(0.02)	(0.01)	(444.6)	(21.41)	(58,904)
		A/R/D	0.312	0.229	0.257	0.188	−2200.4	139.75	14,088,110
			(0.13)	(0.10)	(0.12)	(0.09)	(186.0)	(99.29)	(4,229,160)
		D/R/A	0.312	0.228	0.256	0.186	−2199.3	130.09	14,081,374
			(0.13)	(0.10)	(0.12)	(0.09)	(181.8)	(87.86)	(4,245,441)
Knockdown	100	LG500	0.060	0.028	0.027	0.013	−1535	58.5	5,054,899
			(0.04)	(0.01)	(0.02)	(0.01)	(172.8)	(2.35)	(1312)
		LG1000	0.058	0.029	0.026	0.014	−1529	116.2	10,109,871
			(0.04)	(0.02)	(0.02)	(0.01)	(173.3)	(2.93)	(2647)
		RG500	0.215	0.093	0.142	0.063	−1087	13.3	2,450,303
			(0.09)	(0.03)	(0.06)	(0.02)	(107.2)	(3.2)	(186,656)
		RG1000	0.208	0.092	0.134	0.060	−1098	28.5	5,028,472
			(0.09)	(0.03)	(0.07)	(0.02)	(146.3)	(15.55)	(904,526)
		A/R/D	0.433	0.251	0.349	0.203	−980	3.9	1,271,807
			(0.18)	(0.1)	(0.17)	(0.1)	(67.0)	(3.21)	(472,404)
		D/R/A	0.435	0.252	0.348	0.203	−978	4.0	1,287,779
			(0.18)	(0.1)	(0.17)	(0.1)	(66.7)	(2.98)	(467,089)
	200	RG500	0.113	0.045	0.059	0.024	−2640	50.6	2,499,960
			(0.04)	(0.01)	(0.02)	(0.01)	(187.7)	(17.03)	(521)
		RG1000	0.111	0.044	0.060	0.025	−2631	108.6	4,999,860
			(0.04)	(0.01)	(0.02)	(0.01)	(186.7)	(38.38)	(962)
		A/R/D	0.330	0.190	0.261	0.153	−2182	77.5	11,399,284
			(0.13)	(0.07)	(0.12)	(0.07)	(132.3)	(45.6)	(3,338,144)
		D/R/A	0.332	0.192	0.262	0.154	−2181	79.3	11,469,056
			(0.13)	(0.08)	(0.12)	(0.07)	(131.0)	(47.82)	(3,314,563)

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 associated with both *E. coli* and yeast subnetworks of size 100 and 200 genes. The expression data is generated using linear ODEs, after which a mix of normal and lognormal noise is added by the tool to simulate microarray experimentation noise [43,44]. In total, 160 distinct GNW networks and associated expression data sets are 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. PPV_U and SE_U denote the PPV and SE with respect to an undirected topology, whereas PPV_D and SE_D 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 PPV_U and SE_U, but a false positive in the PPV_D and SE_D 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 6
Summary of search algorithm inferred edges on GNW data.

Type	Genes	Technique	Undirected			Directed			Edges
			TP	FP	FN	TP	FP	FN	
Knockout	100	LG500	7.2	132.3	231.0	3.4	136.1	235.0	139.5
			(4.75)	(52.4)	(119.7)	(2.39)	(53.5)	(121.2)	(54.4)
		LG1000	7.3	133.6	227.2	3.7	137.2	230.9	140.9
			(4.91)	(53.1)	(118.9)	(2.65)	(54.1)	(120.5)	(55.4)
		RG500	24.3	105.9	213.5	15.9	114.4	222.5	130.2
			(11.9)	(24.9)	(111.6)	(8.39)	(25.4)	(115.0)	(27)
		RG1000	23.4	106.1	214.4	15.0	114.5	223.4	129.5
			(12.11)	(25.1)	(112.7)	(8.52)	(25.6)	(115.9)	(27)
		A/R/D	57.5	117.2	179.7	46.1	128.6	192.3	174.7
			(20.33)	(57.3)	(111.5)	(18.53)	(58.4)	(113.9)	(53.9)
		D/R/A	57.9	116.0	179.4	46.8	127.0	191.6	173.8
			(21.24)	(56.8)	(109.9)	(19.16)	(57.6)	(112.9)	(53.8)
	200	RG500	22.2	199.8	518.2	13.0	209.1	527.9	222.0
			(8.49)	(33.6)	(235.8)	(5.36)	(33.5)	(238.8)	(33.6)
		RG1000	21.7	201.0	518.7	12.7	210.0	528.1	222.7
			(7.95)	(32)	(236.5)	(4.87)	(31.9)	(239.5)	(32.1)
		A/R/D	110.9	262.0	427.9	90.7	282.2	450.1	372.9
			(38.28)	(95.8)	(223.7)	(33.29)	(97.0)	(228.9)	(95.1)
		D/R/A	110.5	261.6	428.2	90.0	282.1	450.8	372.1
			(37.97)	(96.6)	(224.0)	(33.31)	(98.0)	(228.8)	(94.7)
Knockdown	100	LG500	6.5	112.7	231.7	3.1	116.1	235.4	119.2
			(4.12)	(43.1)	(119.2)	(2.3)	(43.9)	(120.8)	(44.5)
		LG1000	6.6	118.4	227.8	3.1	121.8	231.5	125.0
			(4.11)	(46.6)	(118.5)	(2.32)	(47.3)	(120.1)	(47.8)
		RG500	21.2	82.8	216.5	14.5	89.5	223.9	104.0
			(9.4)	(19.2)	(113.9)	(6.68)	(19.8)	(116.7)	(20.2)
		RG1000	20.6	83.9	217.3	13.7	90.8	224.8	104.4
			(9.14)	(19)	(115.2)	(6.7)	(19.2)	(117.2)	(18.8)
		A/R/D	52.9	82.1	184.5	42.5	92.5	195.9	135.0
			(18.09)	(48)	(110.1)	(17.01)	(50.0)	(113.4)	(43.1)
		D/R/A	53.0	81.5	184.4	42.4	92.1	196.0	134.4
			(18.16)	(47.2)	(110.2)	(16.84)	(49.3)	(113.6)	(42.6)
	200	RG500	17.7	153.2	389.8	9.8	161.2	398.4	170.9
			(4.39)	(30.6)	(94.0)	(2.49)	(30)	(93.7)	(29.7)
		RG1000	17.4	153.9	390.2	9.9	161.4	398.3	171.3
			(3.55)	(30.1)	(94.2)	(1.89)	(29.7)	(94.5)	(29.3)
		A/R/D	94.0	208.2	444.5	76.0	226.2	464.9	302.2
			(32.47)	(72.9)	(221.8)	(29.55)	(75.8)	(225.9)	(74.6)
		D/R/A	94.7	207.8	443.8	76.3	226.2	464.6	302.5
			(32.58)	(72.9)	(222)	(29.45)	(75.9)	(226.2)	(74.7)

age PPV_U, SE_U, PPV_D, SE_U, 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 *E. coli* and 20 derived from *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

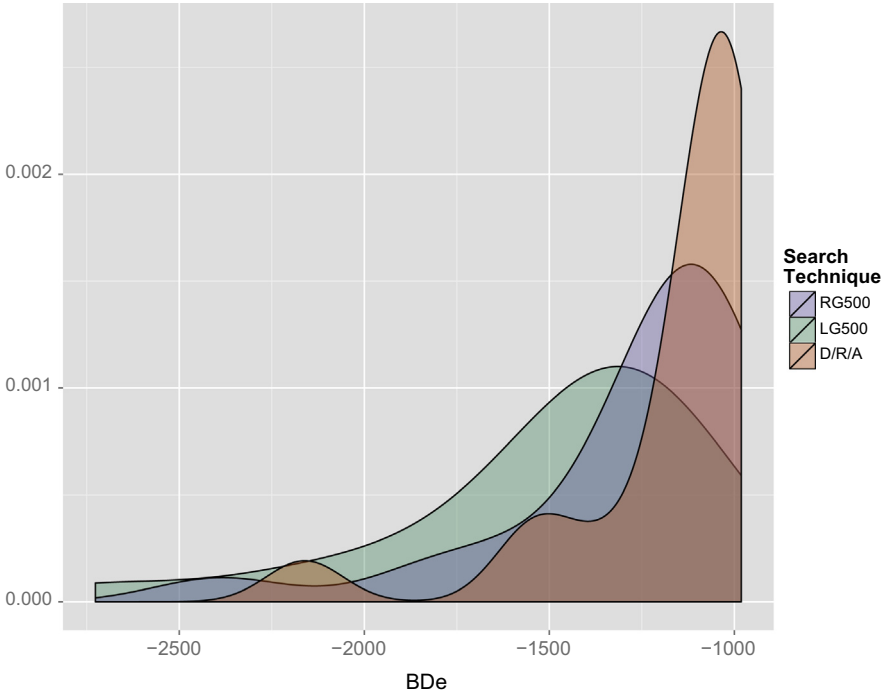

Fig. 6. Estimated densities of BDe scores for 100 gene benchmarks from [5].

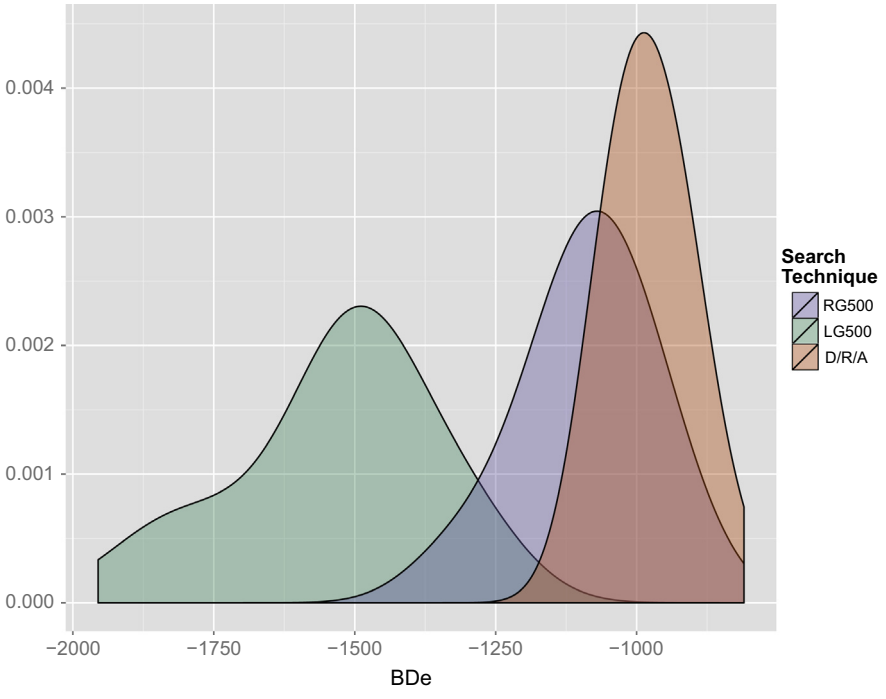

Fig. 7. Estimated densities of BDe scores for 100 gene GNW networks.

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_D values are notably less than the PPV_U values. The local greedy techniques perform poorly with respect to both PPV_U and PPV_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

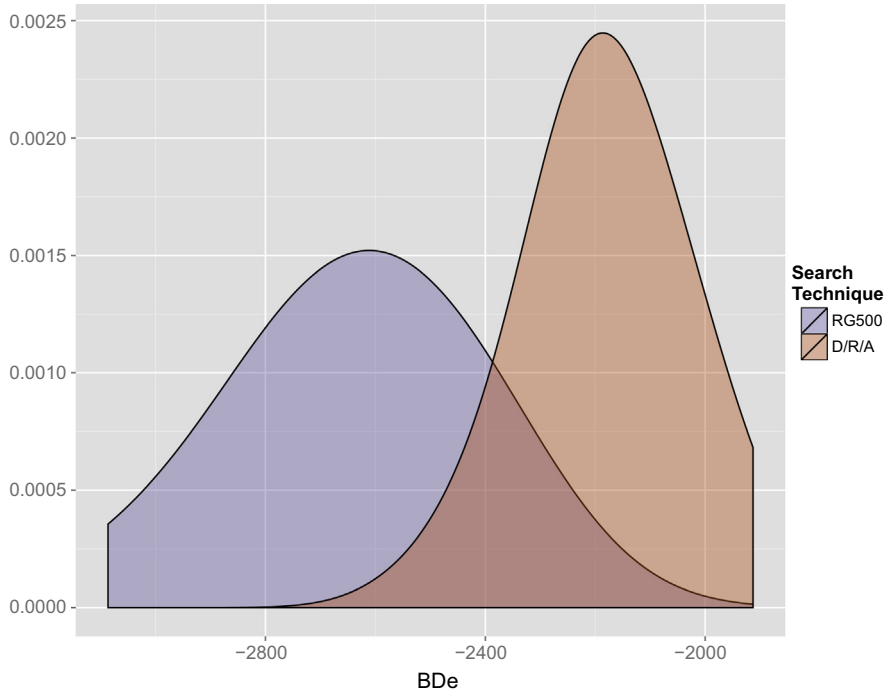

Fig. 8. Estimated densities of BDe scores for 200 gene GNW networks.

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

Genes	Techniques	Undirected			Directed			Edges	BDe	Time
		PPV_U	SE_U	TP	PPV_D	SE_D	TP			
10	A/R/D – RG500	0.06	−0.04	−0.7	−0.04	−0.03	−0.5	−1.5	0.2	−1.51
		(0.11)	(<0.01)	(<0.01)	(0.65)	(0.20)	(0.22)	(<0.01)	(0.62)	(<0.01)
	D/R/A – RG500	0.05	−0.04	−0.6	−0.01	−0.02	−0.4	−1.3	0.9	−1.51
		(0.25)	(0.01)	(0.02)	(0.94)	(0.33)	(0.33)	(<0.01)	(0.17)	(<0.01)
	RG500 – LG500	0.00	0.00	0.00	−0.04	−0.01	−0.1	−0.25	−0.01	−0.45
		(0.91)	(0.98)	(1.00)	(0.49)	(0.51)	(0.49)	(0.17)	(0.75)	(<0.01)
	A/R/D – D/R/A	0.01	0.00	−0.1	−0.03	−0.01	−0.1	−0.2	−0.6	0.00
		(0.34)	(0.58)	(0.43)	(0.37)	(0.51)	(0.54)	(0.21)	(0.28)	(0.58)
100	A/R/D – RG500	0.00	0.03	25.1	0.00	0.02	16.0	30.1	97.5	−14.01
		(0.20)	(<0.01)	(<0.01)	(0.99)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)
	D/R/A – RG500	0.00	0.03	25.1	0.01	0.02	16.0	29.9	100.4	−14.00
		(0.15)	(<0.01)	(<0.01)	(0.70)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)
	RG500 – LG500	0.65	0.00	3.9	0.41	0.00	3.5	−67.8	217.5	−50.71
		(<0.01)	(0.12)	(0.12)	(<0.01)	(0.03)	(0.03)	(<0.01)	(<0.01)	(<0.01)
	A/R/D – D/R/A	0.00	0.00	0.00	−0.01	0.00	−0.1	0.2	−2.9	−0.06
		(0.32)	(1.00)	(1.00)	(0.29)	(0.86)	(0.86)	(0.48)	(0.23)	(0.51)

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 methods shown for the 100 gene 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 of the heuristic approaches were tested with the same input (i.e., gold standard network, expression file, random initial structure) and as such the experiment should be evaluated as a repeated measure. Tables 7 and 8 report the results of the paired *t*-tests among the four techniques for several metrics of interest. The average paired differences (A/R/D − RG500, D/R/A − RG500, RG500 − LG500, and A/R/D − D/R/A) for PPV_U, SE_U, undirected TP, PPV_D, SE_D, directed TP, the number of edges inferred, BDe score, and the computation time in seconds are recorded. Table 7 relates the results on the benchmark data from [5] and Table 8 presents the results from the GNW analysis. A positive value indicates that the first technique obtains a higher value for the given metric. The *p*-value, listed in parentheses below the mean of the differences for each metric, reflect the statistical significance of the value. If a *p*-value is below 0.05 we will refer to the results as statistically significant. For example, consider the PPV_U metric for the 10 gene A/R/D − RG500 paired analysis. The value of 0.06 indicates that the A/R/D was observed to have better PPV_U values in the paired comparison with RG500, however the *p*-value of 0.11 indicates this observed difference is not statistically different from 0.

Tables 7 and 8 confirm that the BDe scores from both A/R/D and D/R/A are statistically superior to the RG500 search heuristic for all tests conducted on the 100 gene or 200 gene sized networks. Table 8 shows that the DNES methods produce statistically better PPV_U, PPV_D, SE_U, and SE_D 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

Table 8
Paired performance differences on GNW data.

Experiment	Genes	Techniques	Undirected			Directed			Edges	BDe	Time
			PPV_U	SE_U	TP	PPV_D	SE_D	TP			
Knockout	100	A/R/D – RG500	0.17	0.17	33.2	0.17	0.15	30.3	44.5	130.4	−8.31
			(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)
		D/R/A – RG500	0.17	0.17	33.6	0.17	0.15	31.0	43.6	132.3	−8.24
			(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)
		RG500 – LG500	0.13	0.07	17.1	0.10	0.05	12.5	−9.2	506.9	−43.55
			(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(0.29)	(<0.01)	(<0.01)
		A/R/D – D/R/A	0.00	0.00	−0.4	0.00	0.00	−0.7	0.85	−1.8	−0.07
			(0.21)	(0.76)	(0.48)	(0.06)	(0.28)	(0.11)	(0.22)	(0.19)	(0.43)
	200	A/R/D – RG500	0.21	0.19	88.7	0.20	0.16	77.7	150.88	688.8	88.29
			(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)
		D/R/A – RG500	0.21	0.18	88.3	0.20	0.16	77.1	150.13	689.9	78.63
			(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)
		A/R/D – D/R/A	0.00	0.00	0.4	0.00	0.00	0.7	0.75	−1.1	9.66
			(0.96)	(0.49)	(0.65)	(0.59)	(0.13)	(0.36)	(0.35)	(0.55)	(0.01)
Knockdown	100	A/R/D – RG500	0.22	0.16	31.7	0.21	0.14	28.0	31.0	107.5	−9.35
			(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)
		D/R/A – RG500	0.22	0.16	31.8	0.21	0.14	27.9	30.48	109.1	−9.29
			(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)
		RG500 – LG500	0.15	0.06	14.63	0.12	0.05	11.43	−15.23	447.44	−45.22
			(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(0.02)	(<0.01)	(<0.01)
		A/R/D – D/R/A	0.00	0.00	−0.1	0.00	0.00	0.1	0.53	−1.6	−0.06
			(0.62)	(0.73)	(0.78)	(0.98)	(0.96)	(0.82)	(0.16)	(0.21)	(0.48)
	200	A/R/D – RG500	0.22	0.18	71.8	0.20	0.16	61.8	118.2	496.1	14.98
			(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(0.12)
		D/R/A – RG500	0.22	0.18	72.5	0.20	0.16	62.0	119.0	493.3	15.49
			(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(0.14)
		A/R/D – D/R/A	0.00	0.00	−0.7	0.00	0.00	−0.3	−0.33	−0.7	−1.80
			(0.10)	(0.08)	(0.12)	(0.50)	(0.35)	(0.46)	(0.74)	(0.64)	(0.27)

search technique. The empirical analysis, which leverages a Bayesian network scoring algorithm, demonstrates that stochastic greedy approaches (i.e., RG500 and RG1000) significantly outperform 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 therefore 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 combination of exploration and exploitation is demonstrated as an effective method for discovering better topologies. Additionally, DNES eliminates the need for random restarts to find good solutions. The systematic switching of neighborhood structures allows the search procedure to escape local optima. DNES finds better quality solutions than the four hill climbing with random restarts implementations analyzed. The current implementation of DNES as a modification to Banjo is available at http://oklahomaanalytics.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 demonstrates 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 incremented from 500 to 1000 restarts, it simply may be that an unrestricted searcher will not be productive. The second potential factor is that as the networks increase in size, the DNES neighborhoods 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. However, 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 simulated 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, consistently 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 neighborhoods 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 quality 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.
[2] P. Mendes, W. Sha, K. Ye, Artificial gene networks for objective comparison of analysis algorithms, Bioinformatics 19 (Suppl. 2) (2003) ii122–ii129.
[3] G. Karlebach, R. Shamir, Modelling and analysis of gene regulatory networks, Nat. Rev. Molec. Cell Biol. 9 (2008) 770–780.
[4] A. Hartemink, Reverse engineering gene regulatory networks, Nature Biotechnol. 23 (5) (2005) 554–555.
[5] M. Bansal, V. Belcastro, A. Ambesi-Impiombato, D. Di Bernardo, How to infer gene networks from expression profiles, Molec. Syst. Biol. 3 (1) (2007) 78.
[6] Y. Huang, I. Tienda-Luna, Y. Wang, Reverse engineering gene regulatory networks, Signal Process. Mag. 26 (1) (2009) 76–97.
[7] T. Ideker, V. Thorsson, R. Karp, Discovery of regulatory interactions through perturbation: inference and experimental design, Pacific Symposium on Biocomputing, vol. 5, 2000, pp. 302–313.
[8] T. Chen, H. He, G. Church, Modeling gene expression with differential equations, Pacific Symposium on Biocomputing, vol. 4, World Scientific, 1999, pp. 29–40.
[9] T.S. Gardner, D. Di Bernardo, D. Lorenz, J.J. Collins, Inferring genetic networks and identifying compound mode of action via expression profiling, Science 301 (5629) (2003) 102–105.
[10] I. Gat-Viks, A. Tanay, D. Raijman, R. Shamir, The factor graph network model for biological systems, in: Research in Computational Molecular Biology, Springer, 2005, pp. 31–47.
[11] J. Rice, Y. Tu, G. Stolovitzky, Reconstructing biological networks using conditional correlation analysis, Bioinformatics 21 (6) (2005) 765–773.
[12] N. Friedman, M. Linial, I. Nachman, D. Pe'er, Using bayesian networks to analyze expression data, J. Comput. Biol. 7 (3–4) (2000) 601–620.
[13] B.-E. Perrin, L. Ralaivola, A. Mazurie, S. Bottani, J. Mallet, F. dAlche Buc, Gene networks inference using dynamic bayesian networks, Bioinformatics 19 (Suppl. 2) (2003) ii138–ii148.
[14] J. Linde, S. Schulze, S. Henkel, R. Guthke, Data-and knowledge-based modeling of gene regulatory networks: an update, EXCLI J. (2015) 346–378.
[15] J. Yu, V. Smith, P. Wang, A. Hartemink, E. Jarvis, Advances to bayesian network inference for generating causal networks from observational biological data, Bioinformatics 20 (18) (2004) 3594–3603.
[16] A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. Favera, A. Califano, ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context, BMC Bioinform. 7 (Suppl. 1) (2006) S7.
[17] B. Ristevski, Overview of computational approaches for inference of micrornamediated and gene regulatory networks, Advances in Computers, vol. 97, Elsevier, 2015, pp. 111–145.
[18] C. de Campos, Q. Ji, Properties of bayesian dirichlet scores to learn bayesian network structures, in: AAAI, 2010, pp. 431–436.
[19] D. Heckerman, D. Geiger, D. Chickering, Learning bayesian networks: the combination of knowledge and statistical data, Machine Learning 20 (3) (1995) 197–243.
[20] G. Cooper, The computational complexity of probabilistic inference using bayesian belief networks, Artif. Intell. 42 (2) (1990) 393–405.
[21] P. Dagum, M. Luby, Approximating probabilistic inference in bayesian belief networks is NP-hard, Artif. Intell. 60 (1) (1993) 141–153.
[22] J. Yu, V. Smith, P. Wang, A. Hartemink, E. Jarvis, Using bayesian network 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 for efficient bayesian network inference, Comput. Biol. Med. 41 (9) (2011) 763–770.
[24] J. Gámez, J. Mateo, J. Puerta, Learning bayesian networks by hill climbing: efficient methods based on progressive restriction of the neighborhood, Data Mining Knowl. Discovery 22 (1–2) (2011) 106–148.
[25] P. Larrañaga, M. Poza, Y. Yurramendi, R. Murga, C. Kuijpers, Structure learning of bayesian networks by genetic algorithms: a performance analysis of control parameters, IEEE Trans. Pattern Anal. Machine Intell. 18 (9) (1996) 912–926.
[26] S. Acid, L. de Campos, J. Castellano, Learning bayesian network classifiers: searching in a space of partially directed acyclic graphs, Machine Learning 59 (3) (2005) 213–235.
[27] L. De Campos, J. Fernandez-Luna, J. Gámez, J. Puerta, Ant colony optimization for learning bayesian networks, Int. J. Approx. Reason. 31 (3) (2002) 291–311.
[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 71 (2–3) (2008) 265–305.
[30] L. Palafox, N. Noman, H. Iba, Reverse engineering of gene regulatory networks using dissipative particle swarm optimization, IEEE Trans. Evol. Comput. 17 (4) (2013) 577–587.

[31] M. Beaumont, B. Rannala, The bayesian revolution in genetics, Nat. Rev. Genet. 5 (4) (2004) 251–261.
[32] A. Hartemink, D. Gifford, Principled computational methods for the validation and discovery of genetic regulatory networks. massachusetts institute of technology, Ph.D. thesis, Ph. D. dissertation, 2001.
[33] C. Needham, I. Manfield, A. Bulpitt, P. Gilmartin, D. Westhead, From gene expression to gene regulatory networks in arabidopsis thaliana, BMC Syst. Biol. 3 (2009) 85.
[34] W. Kunkle, C. Yoo, D. Roy, Reverse engineering of modified genes by bayesian network analysis defines molecular determinants critical to the development of glioblastoma, PLoS ONE 8 (2013) 5.
[35] N. Mladenović, P. Hansen, Variable neighborhood search, Comput. Oper. Res. 24 (11) (1997) 1097–1100.
[36] A. Divsalar, P. Vansteenwegen, D. Cattrysse, A variable neighborhood search method for the orienteering problem with hotel selection, Int. J. Product. Econ. 145 (1) (2013) 150–160.
[37] R. Driessel, L. Mönch, Variable neighborhood search approaches for scheduling jobs on parallel machines with sequence-dependent setup times, precedence constraints, and ready times, Comput. Indust. Eng. 61 (2) (2011) 336–345.
[38] A. Sifaleras, I. Konstantaras, N. Mladenović, Variable neighborhood search for the economic lot sizing problem with product returns and recovery, Int. J. Product. Econ. 160 (2015) 133–143.
[39] P. Hansen, N. Mladenović, J. Brimberg, J. Pérez, Variable neighborhood search, in: M. Gendreau, J.-Y. Potvin (Eds.), Handbook of Metaheuristics, International Series in Operations Research & Management Science, vol. 146, Springer, US, 2010, pp. 61–86.
[40] D. Hurley, H. Araki, Y. Tamada, B. Dunmore, D. Sanders, S. Humphreys, M. Affara, S. Imoto, K. Yasuda, Y. Tomiyasu, K. Tashiro, C. Savoie, V. Cho, S. Smith, S. Kuhara, S. Miyano, D. Charnock-Jones, E. Crampin, C. Print, Gene network inference and visualization tools for biologists: application to new human transcriptome datasets, Nucl. Acids Res. 40 (6) (2012) 2377–2398.
[41] T. Schaffter, D. Marbach, D. Floreano, GeneNetWeaver: in silico benchmark generation and performance profiling of network inference methods, Bioinformatics 27 (16) (2011) 2263–2270.
[42] D. Marbach, T. Schaffter, C. Mattiussi, D. Floreano, Generating realistic in silico gene networks for performance assessment of reverse engineering methods, J. Comput. Biol. 16 (2) (2009) 229–239.
[43] T. Schaffter, D. Marbach, G. Roulet, GeneNetWeaver user manual, September 2016. <http://tschaffter.ch/projects/gnw/downloads/3.1b/gnw-3.1b-user-manual.pdf>.
[44] Y. Tu, G. Stolovitzky, U. Klein, Quantitative noise analysis for gene expression microarray experiments, Proceedings of the National Academy of Sciences of the United States of America 99 (22) (2002) 14031–14036.
[45] E. Adabor, G. Acquaah-Mensah, F. Oduro, Saga: a hybrid search algorithm for bayesian network structure learning of transcriptional regulatory networks, J. Biomed. Inform. 53 (2015) 27–35.