CILAMCE
PANACM
RIO DE JANEIRO 2021

On the Discretization Methods for Single-Cell RNA-Sequencing Data when Inferring Gene Regulatory Networks via Cartesian Genetic Programming

José Eduardo H. da Silva[1], Heder S. Bernardino[1], Itamar L. de Oliveira[1], Alex B. Vieira[1], Helio J.C. Barbosa[1,2]

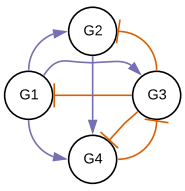
[1]*Universidade Federal de Juiz de Fora, Juiz de Fora, Brazil*
jehenriques@ice.ufjf.br, heder@ice.ufjf.br, alex.borges@ice.ufjf.br, itamar.leite@ice.ufjf.br, hcbm@lncc.br
[2]*Laboratório Nacional de Computação Científica, Petrópolis, Brazil*

Abstract. Gene Regulatory Networks (GRNs) inference from gene expression data (GED) is a hard task and a widely addressed scientific challenge. The sequencing of single-cell RNA (scRNA-seq) allows for the transcriptome exploration at the cellular level and it is attractive for the GRNs inference. GRNs can be represented as Boolean values, in which genes activation and inhibition are presented by logic relationships. Cartesian Genetic Programming (CGP) can be used to evolve GRNs from gene expressions in a binary data form. Therefore, an appropriate discretization technique is important due to its effects on the quality of models. Here, we analyze the performance of ten unsupervised methods for GED discretization when applied to CGP for inferring GRNs. We considered the following discretization methods, based on: statistics, data distribution, ranking, clustering (e.g., k-means and Bikmeans), time series (Transitional State Discrimination), and a method developed by Gallo et al. for data discretization. We also perform a sensibility study of the parameter required by ranking-based methods. We provide a qualitative and quantitative analysis of the discretization approaches in order to obtain a set of methods and parameters that are good for modeling GRNs from scRNA-seq data using CGP.

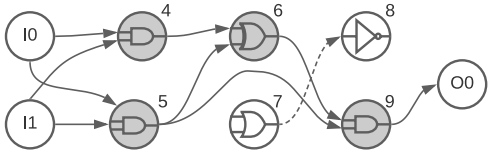
Keywords: Gene Regulatory Network, Cartesian Genetic Programming, scRNA-Seq

1 Introduction

Systems Biology is an interdisciplinary research area that focuses on the interaction between the components of a biological system [1]. Understanding the behavior of organisms at the molecular level depends on understanding what/where/when genes are expressed. Furthermore, gene expression is a complex process regulated at different levels in protein synthesis [2]. All cellular activities are controlled by their genes through a complex network that forms proteins from DNA. Gene expression depends on the relationships between genes in this network, called the Gene Regulation Network (GRN). In the GRNs representation, genes are the network nodes and regulatory regulations are the network edges. Figure 1a shows a GRN with four genes where a direct arrow indicates the activation of the regulatory relationship between two given genes; otherwise, it indicates its inhibition.



(a) Gene Regulatory Network with four genes (G1-G4).



(b) Individual with two primary inputs (I0 and I1) and one output (O0). Continuous lines defines the phenotype, $n_c = n_r = 3$ and $\Gamma = \{\text{AND, OR, NOT, XOR}\}$.

Figure 1. Illustrating a GRN (a) and an individual in CGP (b).

For the inference of GRNs, several data technologies can be used, for instance, RNA-Seq and scRNA-Seq. Bulk RNA-seq technologies have been widely used to study gene expression patterns at population level in the past decade. The advent of single-cell RNA sequencing (scRNA-seq) provides opportunities for exploring gene

expression profiles at the single-cell level. Currently, scRNA-seq has become a favorable choice for studying the key biological questions of cell heterogeneity and the development of early embryos (only include a small number of cells), since bulk RNA-seq mainly reflects the averaged gene expression across thousands of cells [3].

GRNs can be modeled as continuous models in the form of differential equations, or as discrete models, such as Boolean networks. The Boolean-based GRNs considered here model the regulatory relationships in the form of logic functions. This model requires discretized data, and a common scheme for representing the genes' interactions is an alphabet of two symbols $\Gamma = \{0,1\}$, where 0 is inhibition and 1 is activation. Boolean models provide a qualitative measure of gene regulatory mechanisms [4] that, despite its simplicity, can represent through its dynamics, several biologically significant phenomena. The discretization method impacts the quality of the solutions and several techniques have been developed for this purpose, such as those in [5]. However, there is no gold-standard discretization approach. Also, evolutionary computation techniques have been used to model GRNs. Particularly, CGP obtained promising results in the literature [6].

Here, we evaluate 10 unsupervised methods for the discretization of GED when using CGP to infer GRNs discrete models. Among them, ranking-based methods require a user-defined parameter and we also perform a study of the parameter sensibility. The performance is assessed considering three curated models with a proper evaluation framework presented in [7]. The results show that EFD and Median are good choices.

2 Problem Definition

We are interested here in inferring the network topology from a given GED [8]. When considering the GED in the form of time series, the gene expression represents the measurements in each time point. A gene expression dataset is a S x N matrix, where each row vector s (s = 1, ..., S) represents an N-dimensional transcriptome, and each column vector y (y = 1, ..., N) corresponds to an S-dimensional gene profile in the total cell population [9], where N and S stand for the total number of genes and time-points, respectively. The goal of the network inference method is to use this data matrix to predict a set of regulatory interactions between any two genes from the total of N genes. The final output is in the form of a graph with N nodes and a set of edges [9].

Boolean Networks use Boolean Algebra to discover the relationship between genes. Moreover, Boolean networks assume each gene g at a time point t is in one of two states, active or inactive, according to its GED at time t [10]. Therefore, one needs to binarize the data, leading to information loss. In general, the data is discretized in 1 (activation) or 0 (inhibition). Boolean-based models simplify the structure and dynamics of gene regulation. Inferred networks provide a qualitative measure of gene regulatory mechanisms [4]. Also, it is possible to obtain several practical uses, such as the identification of drugs for cancer treatment through the inference of the relationships between genes from experimental data such as gene expression profiles [2].

There are several techniques for measuring the expression of a gene. For instance, single-cell RNA-Sequencing (scRNA-seq) is attractive for GRN inference due to its production of thousands of independent measurements [11] and there are methods that sort cells along "trajectories" describing the development or progress of the cell [12, 13], called pseudotime, which is a measure of how far a cell has moved through biological progress. However, some factors affect the reliability and veracity of the information that can be obtained with scRNA-seq data [14], which can be either biologically-driven, genes not expressing RNA at the time of measurement, or technically-driven, genes expressing RNA, but not at a sufficient level to be detected by sequencing technology [15].

In particular, unlike RNA-seq, the majority of reported expression levels in scRNA-seq are zeros. A matrix of gene expression with many zeros is an issue due to the difficulty of distinguishing and modeling properly the sources of zeros observed, being one of the main challenges for computational analysis [16]. This is known as dropout, which occurs when a gene is observed at a low or moderate expression level in one cell but is not detected in another cell of the same type. In addition, errors may be introduced due to biological variation, such as the stochastic nature of gene expression, environmental niche, and effects created by the cell-cycle [14].

3 Cartesian Genetic Programming

Cartesian Genetic Programming (CGP) is a Genetic Programming technique in which programs are Directed Acyclic Graphs (DAGs) encoded by a matrix of processing nodes, with n_c columns and n_r rows. [17]. The genes are integer values and, for each gene, there are inputs and operation/functions that the node performs. Given a node in the matrix, the nodes at its left side can be used as inputs. There is a user-defined parameter l_b (levels-back) that limits the number of columns at the left side where inputs can be selected to constrain the connectivity of the graph. Also, the genotype contains nodes that contribute directly to the output, called active nodes, and those that do not, the inactive nodes. The phenotype is composed only of the active nodes and the genotype-phenotype mapping is done by recursively determining the nodes that contribute to each output, starting on the output and ending on the primary inputs. The function set is user-defined and problem-dependent. For example, logic functions or gates are

used when designing digital circuits. Figure 1b presents a CGP individual with two primary inputs and one output. The functions are logical ones. Grey nodes are active and white nodes, inactive.

The most common search technique used in CGP is the $(1 + \lambda)$ Evolutionary Strategy (ES) [17], where λ is the number of new solutions generated at each iteration. In this case, the best individual, the one with more matches concerning its truth-table, between the parent and the λ new solutions is selected for the next generation.

CGP normally uses only mutation to generate new individuals [17]. Here, we consider the use of Single Active Mutation (SAM) [18] which reduces the wasted objective function evaluations, where (i) one node and one of its elements (inputs or function) are selected at random, and (ii) its value is changed to another valid value. Steps (i) and (ii) are repeated until an active node is modified. The general CGP's procedure is shown in Algorithm 1.

Algorithm 1 CGP's $(1 + \lambda)$-ES general procedure [17].

1: Randomly generate the individual population and select the fittest individual as parent
2: **while** the maximum number of evaluations allowed is not reached **do**
3: Generate λ new individuals (by mutating the parent individual) and evaluate them
4: Select as parent the best individual among the $(1 + \lambda)$ current individuals
5: **end while**

4 Discretization Methods

For our proposal, we consider the data from three curated models provided in [7]. Curated models were generated considering four published Boolean models: mammalian cortical area development (mCAD) [19], ventral spinal cord (VSC) development [20], and hematopoietic stem cell (HSC) differentiation [21]. These datasets were simulated using BoolODE [7] in order to obtain data in the form of scRNA-seq. BoolODE was also used to create ten different datasets with 2,000 cells for each model. For each dataset, three versions of dropout are available with $q \in \{0\%, 50\%, 70\%\}$. For each dropout category, 10 datasets were generated. Then, for each curated problem, 30 datasets are available. Pseudotimes were computed using Slingshot [22].

We considered two scenarios. In the first, we try to infer the GRN considering the original data. In the second, the data is initially ordered according to the pseudotime information of the cells. Thus, we have a distribution of gene expression points over time. To remove or soften the effects of technical and biological variations and to obtain a function that represents gene expression over time, we use data approximation using cubic smoothing splines implemented in Python and distributed through the library *csaps*[1]. The smoothing value was evaluated in [0, 0.99] with a step of 0.05. Low smoothing values can cause underfitting while high ones lead to overfitting.

There are several approaches for discretizing GED, and here we evaluate 10 unsupervised methods for the discretization of GED. For further reading, we refer to [5].

Considering a gene expression matrix A', where the rows are the genes and the columns are the time points, the simplest method is to discretize the GED considering two levels of discretization with $a_{ij} = 1$, when $a'_{ij} \geq \delta$, and 0, otherwise, where a_{ij} is the discretized value and a'_{ij} is the real gene expression value. The value δ can be defined as the mean or the median of the A' row. Also, one can consider δ as some sort of expression considering a fixed proportion X regarding the maximum value of the A' row (known as Max -X%Max).

Furthermore, one can perform discretization based on ranking. Assuming that the expression values are sorted in decreasing order on a list L, the simplest approach is to assign the first X% values of L to 1, whereas the other values are assigned to 0. This approach is known as Top%X [23, 24]. Also, one can use 'Equal Width Discretization' (EWD) in which the difference between the maximum and the minimum values of A' row is divided into k intervals of equal width. Another related approach is based on the equal frequency principle. This method, known as 'Equal Frequency Discretization' (EFD) [25], considers a given number k of symbols into which the expression values will be discretized. Then L is split into k segments of length $|L|$/k containing the same number of data points per symbol, thus assigning the k discrete states according to the decreasing order of the segments.

Other approaches that deal with the discretization of GED are based on clustering. The way to achieve this is to consider each value a'_{ij} of the GED A' as an element of a single-dimensional space Ω. Then, a clustering algorithm is applied to the S elements of X that correspond to a specific 'data scope' (a gene profile, a column profile, or a matrix profile) to obtain groups of values, where the values belonging to the same group are assigned to the same discrete state. The groups are calculated by maximizing the similarity within the elements of each cluster, while minimizing this value among elements in different clusters. A common quality metric for the clusters is the WCSS (Within-Cluster Sum of Squares), defined as $WCSS(D) = \sum_{a'_{ij} \in [p_0, p_1]} |a'_{ij} - \mu_0|^2 + \sum_{r=1}^{k-1} \sum_{a'_{ij} \in (p_r, p_{r+1}]} |a'_{ij} - \mu_r|^2$, where μ_r is the mean of the $a'_{ij} \in (p_r, p_{r-1}]$. Basically, the WCSS is the sum

[1] https://csaps.readthedocs.io/en/latest/index.html

of the squared Euclidean distance between the elements within a cluster and the mean of that cluster, where lower values mean higher similarities between the elements of the clusters.

A widely used algorithm for this task is the k-means clustering [26]. The k-means uses the Squared Euclidean distance as a similarity measure, trying to yield a partition of elements with the least WCSS, as before. The main steps of the algorithm can be summarized as follows: (i) the algorithm takes a set of points S and a fixed integer k as input, (ii) splits S into k subsets by choosing a set of k initial centroid points, where the elements of S are grouped regarding their nearest centroid to form the clusters, and (iii) recalculate the centroids from the elements within the clusters. Steps (ii) and (iii) are iterated until some stopping criterion is met (usually convergence). The choice of the initial centroids is a key aspect, as it influences the final structure of the partition. Commonly, it is started with random centroids and a different clustering of S may be obtained every time this method is run [27].

When dealing with GED, the most common approach is to use the k-means algorithm to discretize either the gene expression profiles or the condition expression profiles [28, 29], such as Bidirectional K-means (Bikmeans) [28]. Bikmeans uses both the clustering of gene profiles (expression level) and column profiles (time information of experimental condition) using the k-means algorithm [30]. That is, for a given "level of discretization" of k, the algorithm identifies $(k+1)$ clusters for the gene profiles and the condition profiles, independently [5].

Considering an $N \times M$ expression data matrix $\mathbf{E}$, where N represents the expression values and M the time points, Bikmeans uses K-means [30] clustering, which divides **E(n,:)**, so that adjacent expression values of gene n are divided into the same interval. Also, Bikmeans uses Cokmeans (K-means for columns) where the matrix **E(:,m)** is divided into k intervals so that adjacent expression values at time point m are divided into same interval [28]. Finally, Bikmeans computes $(k+1)$-means clusters for the expression values and time points, independently, giving every expression value two possible discrete states, a'_{ij}: one for the expression value (a^e_{ij}), and one for the time point, a^t_{ij}, with $1 \leq a^e_{ij} \leq k+1$, $1 \leq a^t_{ij} \leq k+1$. Then, the discrete state a_{ij}, with $1 \leq a_{ij} \leq$ k is assigned to a'_{ij} if $(a_{ij})^2 \leq a^g_{ij} a^c_{ij} < (a_{ij}+1)^2$. Considering $k = 3$, Table 1 shows an example of possible discrete states for a_{ij}. In this case, k-means is performed $N + M$ times as both expression values and time points are clustered [5].

When using GED in the form of variation between time points, the simplest approach is called Transitional State Discrimination (TSD) [31]. TSD standardizes the values using z-scores with a normal distribution [5] as $a_{ij} = 1$, $a'_{ij} - a'_{i(j-1)} \geq 0$, and 0, $a'_{ij} - a'_{i(j-1)} < 0$, where a_{ij} is the discretized value and a'_{ij} is the real data values. The values a'_{ij} generated by this transformation are then used to calculate the discrete state a_{ij}. According to this discretization, a_{ij} equals 1 (active) when the observed values increase, and 0 (inactive), otherwise.

Finally, we consider a discretization approach developed by [32], where the discretization of a gene *i* can be defined as $\min_{S_1,S_2 \subset S}$ (var(S_1) + var(S_2)), where S is the set of sample values for the gene *i*, $S_1 \cap S_2 = \emptyset$, $S_1 \cup S_2 = S$, $|S_1| > 1$ and $|S_2| > 1$, var(S_1) and var(S_2) are the variance of S_1 and S_2 respectively, and S_1 and S_2 represent the two expression states for the gene *i*. The samples of the gene *i* are divided into the two sets that have the minimum sum of their variances. The cardinality of S_1 and S_2 is required to be greater than one in order to avoid the effects of a possible outlier in the samples. Thus, when the samples of a gene *i* are separated in a partition that violates this restriction, gene *i* is no longer considered in the inference process for the current datasets.

Table 1. Discretization via Bikmeans.

		K-means			
		1	2	3	4
Cokmeans	1	1×1 = 1 → a_{ij} = 1	a_{ij} = 1	a_{ij} = 1	a_{ij} = 2
	2	2×1 = 2 → a_{ij} = 1	a_{ij} = 2	a_{ij} = 2	a_{ij} = 3
	3	3×1 = 3 → a_{ij} = 1	a_{ij} = 2	a_{ij} = 3	a_{ij} = 3
	4	4×1 = 4 → a_{ij} = 2	a_{ij} = 3	a_{ij} = 3	a_{ij} = 3

Table 2. Problems used in the experiments.

Problem	**#Genes**	**#Pseudotimes**
HSC	11	4
mCAD	5	2
VSC	8	5

5 Computational Experiments

Experiments were conducted to analyze the performance of different discretization approaches when applied to CGP as inference algorithm using benchmark scRNA-Seq time-series data. The results are obtained using the BEELINE framework evaluation [7], which considers the area under the precision-recall curve (AUPRC) and the area under the receiver operating characteristic curve (AUROC) values as metrics.

An excellent model has AUC, both for ROC and PRC, close to 1, which means it has a good separability. For many real-world datasets, particularly medical datasets, the fraction of positive cases is often less than 0.5, meaning that AUPRC has a lower baseline value than AUROC [33].

Also, we use performance profiles (PPs) [34] to analyze the relative performance of the methods. In PPs,

the probability that the performance of a given method is within a factor $\tau > 1$ of the best one is $\rho_s(\tau)$, and one can extract: (i) the approach that obtained the best results for most problems (largest $\rho(1)$), (ii) the most reliable approach (smaller τ such that $\rho(\tau) = 1$), and (iii) the best overall performance (largest area under the PPs curves).

The problems are presented in Table 2. All discretizations were performed using Gene Expression Data Pre-Processing Tool (GEDPROTOOLS)[2] and our implementations are available[3].

As each pseudotime gives information about one possible cell trajectory, for each pseudotime one GRN is inferred. Then, the final GRN is given by merging the partial GRNs discovered for each pseudotime. Duplicated regulatory relationships are removed. Only stronger regulatory relationships are kept in this case. The final GRN is evaluated and the regulatory relationships are ranked from the stronger to the weaker. For all experiments we consider n_c = l_b = 100, n_r = 1 and the maximum number of evaluations of the objective function is 50,000.

Preliminary experiments were carried out considering different parameters for the Top%X and Max -X%Max methods. For the first, values in [25, 60] with a step of 5% were considered. For Max -X%Max, the reference value is 54% [23] and, thus, we analyzed {50, 54, 60}. In addition, experiments using data with and without spline were performed. In general, data with spline provided better results of CGP. Thus, the remaining analysis considers the results obtained with spline and with the best parameters obtained for Top (50%) and Max (50%). The analyses to obtain these conclusions are available in the supplementary material.

Figure 2 presents the results for AUPRC and AUROC for HSC, mCAD, and VSC. The boxplots presented in Figure 2a and 2b show that Bikmeans performed better for HSC. Also, even with 70% dropout, CGP with Bikmeans performed well. However, for mCAD and VSC, EFD showed better results in most cases. All discretization approaches showed great variance, independently of the dropout rate. For all problems, using Mean as discretization method, the results obtained with 50% and 70% dropout are the same. In general, dropouts did not affect substantially the performance of CGP, different from related in the literature. This may occur because the dropouts are considered don't care situations, facilitating CGP in obtaining feasible solutions.

Figures 2g and 2h show PPs for AUPRC and AUROC, respectively. For AUPRC, one concludes that (i) Median found the best results for most problems (largest $\rho(1)$), (ii) KmeansRow is the most reliable approach (smaller τ such that $\rho(\tau) = 1$), and (iii) EFD obtained the best overall performance (largest area under the performance profiles curves). For AUROC, one concludes that (i) Median and EFD are approaches that found the best results for most problems, (ii) Median is the most reliable approach, and (iii) EFD obtained the best overall performance.

Thus, one can conclude that EFD obtained the best overall performance in both AUPRC and AUROC. However, Median is a good choice and obtained the second-best overall performance.

6 Concluding Remarks and Future Works

We evaluated here 10 unsupervised methods for the discretization of gene expression data when applied to CGP to infer GRNs discrete models. Among them, ranking-based methods require a user-defined parameter and we performed a study of the sensibility of these parameters, where we concluded that Top%X and Max -X%Max performed better with 50%. Also, we evaluated the effect of using spline to smooth the data and the results show that CGP is able to find more feasible solutions and the results for AUPRC and AUROC are better when using spline. Bikmeans performed better for the problem HSC. For mCAD and VSC, EFD obtained better results. However, all discretization methods showed high variance. In general, PPs showed that EFD provided the best overall performance concerning both AUPRC and AUROC. As future work, we intend to extend this analysis to real human and mice datasets and assess the performance of CGP using EFD in our previous works.

Acknowledgements We thank the support provided by CAPES, CNPq(312337/2017-5), FAPERJ, FAPESP, FAPEMIG, UFJF, and Amazon AWS.

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

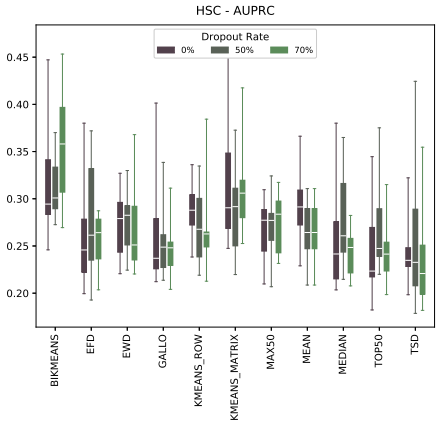
[1] R.-S. Wang, A. Saadatpour, and R. Albert. Boolean modeling in systems biology: an overview of methodology and applications. *Physical biology*, vol. 9, n. 5, pp. 055001, 2012.

[2] M. N. McCall. Estimation of gene regulatory networks. *Postdoc journal: a journal of postdoctoral research and postdoctoral affairs*, vol. 1, n. 1, pp. 60, 2013.

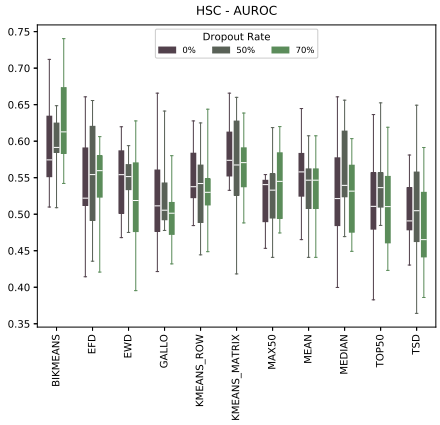
[3] G. Chen, B. Ning, and T. Shi. Single-cell rna-seq technologies and related computational data analysis. *Frontiers in genetics*, vol. 10, pp. 317, 2019.

[2] http://lidecc.cs.uns.edu.ar/files/gedprotools.zip

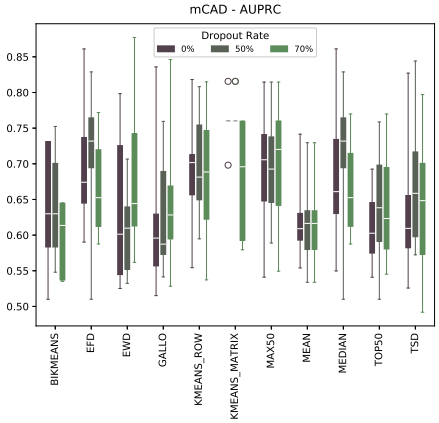
[3] https://github.com/ciml/cilamce2021_cgp-grn-discretization



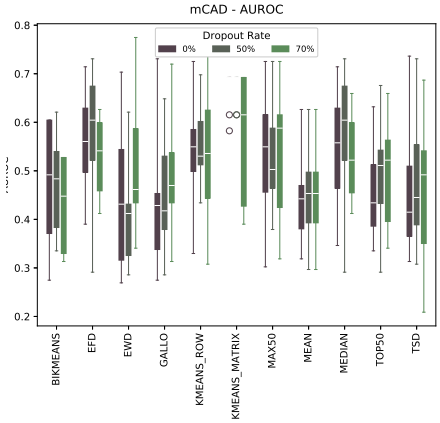
(a) Results for HSC AUPRC



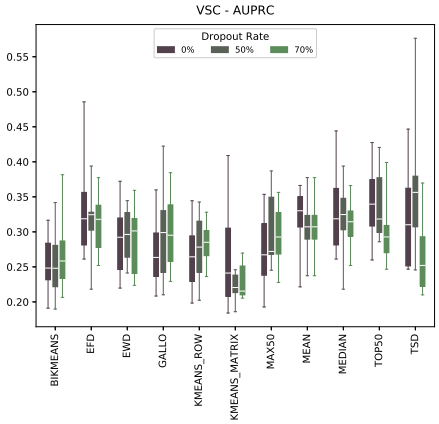
(b) Results for HSC AUROC



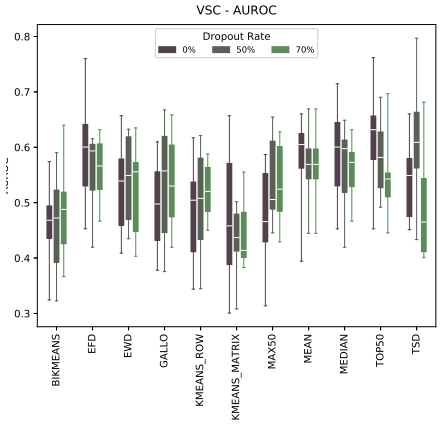
(c) Results for mCAD AUPRC



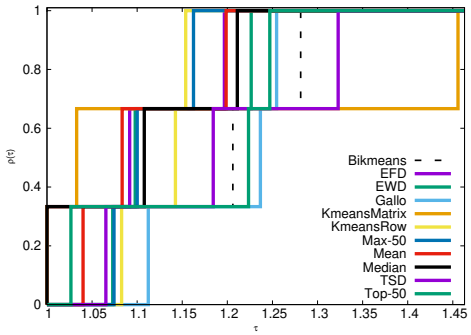
(d) Results for mCAD AUROC



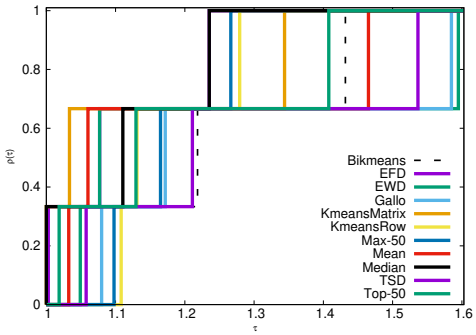
(e) Results for VSC AUPRC



(f) Results for VSC AUROC



(g) PPs using AUPRC



(h) PPs using AUROC

Figure 2. Results obtained for all scenarios (a-f) and PPs considering AUPRC (g) and AUROC (h).

[4] G. Sanguinetti and others. Grn inference: an introductory survey. In *GRN*, pp. 1–23. Springer, 2019.
[5] C. A. Gallo, R. L. Cecchini, J. A. Carballido, S. Micheletto, and I. Ponzoni. Discretization of gene expression data revised. *Briefings in bioinformatics*, vol. 17, n. 5, pp. 758–770, 2015.
[6] da J. E. H. Silva and H. S. Bernardino. Cartesian genetic programming with crossover for designing combinational logic circuits. *Proc. of 7th Brazilian Conference on Intelligent Systems*, vol. , pp. 145–150, 2018.
[7] A. Pratapa, A. P. Jalihal, J. N. Law, A. Bharadwaj, and T. Murali. Benchmarking algorithms for gene regulatory network inference from single-cell transcriptomic data. *Nature methods*, vol. 17, n. 2, pp. 147–154, 2020.
[8] A. Aalto, L. Viitasaari, P. Ilmonen, L. Mombaerts, and J. Gonçalves. Gene regulatory network inference from sparsely sampled noisy data. *Nature communications*, vol. 11, n. 1, pp. 1–9, 2020.
[9] S. Chen and J. C. Mar. Evaluating methods of inferring gene regulatory networks highlights their lack of performance for single cell gene expression data. *BMC bioinformatics*, vol. 19, n. 1, pp. 1–21, 2018.
[10] M. Banf and S. Y. Rhee. Computational inference of gene regulatory networks: approaches, limitations and opportunities. *Bioch. et Biophysica Acta (BBA)-Gene Regulatory Mechanisms*, vol. 1860, n. 1, pp. 41–52, 2017.
[11] S. Liu and C. Trapnell. Single-cell transcriptome sequencing: recent advances and remaining challenges. *F1000Research*, vol. 5, 2016.
[12] L. Haghverdi, M. Büttner, F. A. Wolf, F. Buettner, and F. J. Theis. Diffusion pseudotime robustly reconstructs lineage branching. *Nature methods*, vol. 13, n. 10, pp. 845, 2016.
[13] X. Qiu, Q. Mao, Y. Tang, L. Wang, R. Chawla, H. A. Pliner, and C. Trapnell. Reversed graph embedding resolves complex single-cell trajectories. *Nature methods*, vol. 14, n. 10, pp. 979, 2017.
[14] B. Hwang, J. H. Lee, and D. Bang. Single-cell rna sequencing technologies and bioinformatics pipelines. *Experimental & molecular medicine*, vol. 50, n. 8, pp. 1–14, 2018.
[15] S. C. Hicks, F. W. Townes, M. Teng, and R. A. Irizarry. Missing data and technical variability in single-cell rna-sequencing experiments. *Biostatistics*, vol. 19, n. 4, pp. 562–578, 2018.
[16] T. S. Andrews, V. Y. Kiselev, D. McCarthy, and M. Hemberg. Tutorial: guidelines for the computational analysis of single-cell rna sequencing data. *Nature Protocols*, vol. , pp. 1–9, 2020.
[17] J. F. Miller. Cartesian genetic programming. *CGP*, vol. , pp. 17–34, 2011.
[18] B. Goldman and Punch. Reducing wasted evaluations in cgp. In *EuroGP*, pp. 61–72. Springer, 2013.
[19] C. E. Giacomantonio and G. J. Goodhill. A boolean model of the gene regulatory network underlying mammalian cortical area development. *PLoS Comput Biol*, vol. 6, n. 9, pp. e1000936, 2010.
[20] A. Lovrics and Others. Boolean modelling reveals new regulatory connections between transcription factors orchestrating the development of the ventral spinal cord. *PloS one*, vol. 9, n. 11, pp. e111430, 2014.
[21] J. Krumsiek, C. Marr, T. Schroeder, and F. J. Theis. Hierarchical differentiation of myeloid progenitors is encoded in the transcription factor network. *PloS one*, vol. 6, n. 8, pp. e22649, 2011.
[22] K. Street, D. Risso, R. B. Fletcher, D. Das, J. Ngai, N. Yosef, E. Purdom, and S. Dudoit. Slingshot: cell lineage and pseudotime inference for single-cell transcriptomics. *BMC genomics*, vol. 19, n. 1, pp. 1–16, 2018.
[23] S. C. Madeira and A. L. Oliveira. An evaluation of discretization methods for non-supervised analysis of time-series gene expression data. *INESC-ID Technical Report*, vol. 42, pp. 2005, 2005.
[24] S. Garcia, J. Luengo, J. A. Sáez, V. Lopez, and F. Herrera. A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning. *IEEE Trans. Knowl. Data Eng.*, vol. 25, n. 4, pp. 734–750, 2012.
[25] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *Machine learning proceedings 1995*, pp. 194–202. Elsevier, 1995.
[26] J. MacQueen and others. Some methods for classification and analysis of multivariate observations. In *Proc. of the Berkeley symposium on mathematical statistics and probability*, volume 1, pp. 281–297, 1967.
[27] M. Elloumi and A. Y. Zomaya. *Biological Knowledge Discovery Handbook: Preprocessing, Mining and Postprocessing of Biological Data*, volume 23. John Wiley & Sons, 2013.
[28] Y. Li, L. Liu, X. Bai, H. Cai, W. Ji, D. Guo, and Y. Zhu. Comparative study of discretization methods of microarray data for inferring transcriptional regulatory networks. *BMC Bioinf.*, vol. 11, n. 1, pp. 520, 2010.
[29] P. Mahanta, H. A. Ahmed, J. K. Kalita, and D. K. Bhattacharyya. Discretization in gene expression data analysis: a selected survey. In *Intl. Conf. on Comp. Science, Eng. and Information Tech.*, pp. 69–75, 2012.
[30] K. Krishna and M. N. Murty. Genetic k-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 29, n. 3, pp. 433–439, 1999.
[31] C. S. Möller-Levet, K. Chu, and O. Wolkenhauer. Dna microarray data clustering based on temporal variation: Fcv with tsd preclustering. *Applied Bioinformatics*, vol. 2, n. 1, pp. 35–45, 2003.
[32] C. A. Gallo, J. A. Carballido, and I. Ponzoni. Discovering time-lagged rules from microarray data using gene profile classifiers. *BMC bioinformatics*, vol. 12, n. 1, pp. 1–21, 2011.
[33] R. Draelos. Measuring performance: Auprc and average precision, 2019.
[34] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Program.*, vol. 91, n. 2, pp. 201–213, 2002.