Designing Distributed Cell Classifier Circuits Using a Genetic Algorithm

Melania Nowicka[1,2(✉)] and Heike Siebert[1]

[1] Freie Universitaet, 14195 Berlin, Germany
m.nowicka@fu-berlin.de
[2] Max Planck Institute for Molecular Genetics, 14195 Berlin, Germany

Abstract. Cell classifiers are decision-making synthetic circuits that allow in vivo cell-type classification. Their design is based on finding a relationship between differential expression of miRNAs and the cell condition. Such biological devices have shown potential to become a valuable tool in cancer treatment as a new type-specific cell targeting approach. So far, only single-circuit classifiers were designed in this context. However, reliable designs come with high complexity, making them difficult to assemble in the lab. Here, we apply so-called Distributed Classifiers (DC) consisting of simple single circuits, that decide collectively according to a threshold function. Such architecture potentially simplifies the assembly process and provides design flexibility. We present a genetic algorithm that allows the design and optimization of DCs. Breast cancer case studies show that DCs perform with high accuracy on real-world data. Optimized classifiers capture biologically relevant miRNAs that are cancer-type specific. The comparison to a single-circuit classifier design approach shows that DCs perform with significantly higher accuracy than individual circuits. The algorithm is implemented as an open source tool.

Keywords: Synthetic biology · Boolean modeling · Genetic algorithms · miRNA profiling · Cell classifiers · Cancer

1 Introduction

Synthetic biology has shown its immense potential in recent years in a wide array of applications. This is particularly true for the medical field, where synthetic biological systems are developed for versatile employment from diagnostics to treatment [24,28]. Research in design and construction of cell classifier circuits touches on both these areas. Cell classifiers are molecular constructs capable of sensing certain markers in the environment, processing the input and reacting with a signal-specific output. A prime example for this are miRNA-based classifiers that distinguish cell states, e.g., as healthy or diseased, based on their miRNA expression profiles applying boolean logic (Fig. 1A) [15,26]. These circuits can be delivered to cells on plasmids or viral vectors and trigger the production of a desired output, e.g., a toxic compound causing cell apoptosis in diseased cells (Fig. 1B).

L. Bortolussi and G. Sanguinetti (Eds.): CMSB 2019, LNBI 11773, pp. 96–119, 2019.
https://doi.org/10.1007/978-3-030-31304-3_6

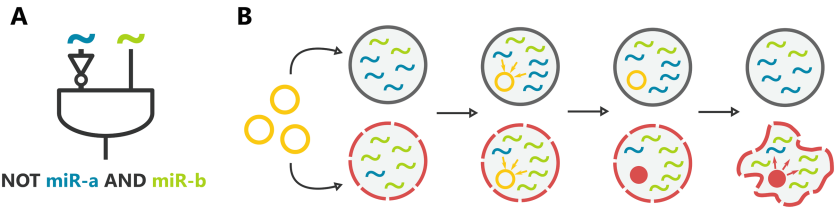

Fig. 1. (A) An exemplary boolean design of a two miRNA-input cell classifier. (B) A schema showing two types of cells, healthy (solid line) and diseased (dashed line). The classifiers are delivered to the cells, sense the internal input levels and respond with respect to a given cell condition.

A variety of different approaches to designing synthetic circuits is available [12,17,25]. However, to confront many application-derived limitations, circuit designs must be often tailored to rigorous specifications. Since cell classifiers must be feasible to implement in the lab, many constraints are posed on the building blocks of these circuits that need to be encoded in the design problem. So far, only a few different methods for designing single-circuit classifiers were described [1,15]. Mohammadi et al. proposed two different heuristic approaches [15]. The procedure performing with the highest accuracy in terms of sample classification allows to optimize a classifier's topology using a mechanistic model of the circuit and a predefined set of biochemical parameters. Another approach was presented by Becker et al. [1]. The authors propose a method for finding globally optimal classifiers represented by boolean functions based on binarized miRNA expression data. To search through the entire space of solutions in a short time frame the authors apply logic solvers. Becker et al. compare their results to the previously mentioned state-of-the-art method demonstrating significant improvement in binary classification of presented classifiers [1].

While this research shows that theoretically single-circuit classifiers can perform such classification tasks [1,15], there is a number of challenges for the approach in application. Depending on the heterogeneity of the data, to obtain a clear-cut classification often a circuit of high complexity is needed. Generally, the cost both in time and money for classifier circuit construction in the lab goes up the larger and more complex the circuit architecture gets, quickly becoming not feasible at all [15]. A further problem is the robustness needed for reliable performance when faced with uncertainty and noise in signals and wide ranging possibilities for perturbations of the classifier functionality in natural environments. To address these issues the principles of distributed classification, as inherent in many natural systems such as the immune system and shown to be an effective strategy, e.g., in machine learning, can be exploited [19,22]. Here, the idea is to design a set of different classifier circuits, also called distributed classifier, that perform classification in an integrated manner. Such a set can consist of rather simple classifiers that still perform more accurately than a complex single circuit classifier, since the individual classification results are aggregated which compensates for individual mistakes. A theoretical design of

such a distributed classifier based on synthetic gene circuits was presented by Didovyk et al. [3]. The classifier is optimized by training a starting population of simple circuits on the available data similarly to machine learning algorithms, i.e., by presenting learning examples and successively removing low-performance circuits. While this work considers only a quite specific scenario being designed for bacterial cell cultures, it highlights the potential of the underlying idea of using distributed classifiers.

Here, we adapt the distributed classifier approach proposed by Didovyk et al. [3] to the problem of cell classifier design. We define a *Distributed Classifier* (DC) as a set of single-circuit classifiers that decide collectively based on a threshold function. Biologically, the threshold may correspond to a certain concentration of the drug that allows to treat the cells or fluorescent marker allowing to classify the cell type [3,14,15]. According to Mohammadi et al. [15] such threshold manipulation may be achieved by changing the biochemical parameters of a circuit model. Due to the high complexity of the problem, we apply a heuristic approach to design and optimize DCs, namely, a genetic algorithm (GA). GAs are evolution-inspired metaheuristics that allow to optimize populations of individuals [13]. Such evolutionary approaches were successfully applied to various biological questions [11], e.g., design of synthetic networks and, in particular, design of single-circuit classifiers [15,25]. Due to the high flexibility of GAs in terms of design and parameters, the algorithm may be efficiently adapted to the distributed classifier problem.

In this article, we illustrate the potential of distributed classifiers in application, in particular, in cancer cell classification. The following section contains preliminaries including the definition of a single-circuit and distributed classifier. Section 3 describes the architecture of the proposed genetic algorithm for the design and optimization of DCs. In Sect. 4 we present case studies performed on real-world breast cancer data and compare the results with a single-circuit design method proposed by Becker et al. [1]. Finally, we discuss the distributed classifier performance and comment on potential future work.

2 Preliminaries

In this section we describe the data we employ to designing classifiers, introduce single-circuit and distributed classifiers and propose binary classification measures that allow to evaluate their performance.

2.1 miRNA Expression Data

The proposed method is a boolean approach and utilizes binarized and annotated data. While our focus is on miRNA expression profiles, the approach can naturally be applied to any data set of the format introduced below.

In cancer research, differentially expressed miRNAs provide a valuable source of information about tumor development, progression and response to a therapy [8,9]. Thus, dysregulated miRNAs have been considered as potential biomarkers

for cancer diagnosis and treatment. One of the approaches allowing to distinguish up- and down-regulated miRNAs is discretization of the expression data into a finite number of states. Discretization provides clear and interpretable information about the miRNA behaviour and makes the learning process from the data more efficient [6]. However, the procedure is also related to a potential information loss. We comment on this issue in Sect. 5.

Table 1. A miRNA expression data set.

ID	Annots	*miR-a*	*miR-b*	*miR-c*
1	0	0	1	0
2	0	0	1	0
3	1	1	0	0
4	1	0	0	0
5	1	1	0	0

We define a data set $D = (S, A)$ as a finite set of samples $S \subseteq \{0,1\}^m$, where $m \in \mathbb{N}$ is the number of miRNAs and $A : S \longrightarrow \{0,1\}$ is sample annotation. Presented as a table, the first column includes unique sample IDs and the second the annotation of samples (Annots), where 0 is a label assigned to negative class samples (healthy) and 1 to positive (cancerous). The following columns are miRNA expression profiles describing the miRNA regulation among the samples. miRNAs are binarized into two states: up- (1/positive) and down-regulated (0/negative), according to a given threshold. An example of a data set is presented in Table 1. A miRNA is non-regulated if for every sample its state is either 0 or 1 (e.g., Table 1, *miR-c*). Some miRNAs can perfectly separate the samples into the two categories implied by the annotation (e.g., Table 1, *miR-b*).

2.2 Single-Circuit Classifier

A single-circuit cell classifier may be represented by a boolean function $f : S \longrightarrow \{0,1\}$. To make a classifier feasible to construct in the lab additional constraints must be imposed on the function. We adopt here the constraints introduced by Mohammadi et al. [15]. Accordingly, the function should be given in *Conjunctive Normal Form* (CNF), i.e., a conjunction of clauses where each clause is a disjunction of negated (negative) or non-negated (positive) literals. Here, the literals correspond to the miRNAs and clauses to the gates. It may consist of: (i) negative literals only in 1-element clauses (NOT gates) (ii) at most 3 positive literals per clause (OR gate) (iii) up to 10 literals (miRNAs) and up to 6 clauses (gates) in total (iv) including at most 4 NOT gates and 2 OR gates. A circuit topology presented as a CNF satisfying the above-mentioned constraints directly corresponds to the biological model of the circuit employed by Mohammadi et al. [15]. An example of a classifier is presented below.

$$\neg miR\text{-}a \wedge (miR\text{-}b \vee miR\text{-}c) \tag{1}$$

The function should output 1/True in case of cancerous and 0/False in case of healthy cells. The example function presented in Eq. 1 classifies a cell as positive/1 if *miR-a* is down-regulated and at least one of the other miRNAs (*miR-b* or *miR-c*) is up-regulated.

2.3 Distributed Classifier

Here, we introduce a concept of *Distributed Classifier* (*DC*) for the cell classification problem. A DC is a finite set $DC = \{f_1, ..., f_c\}$, where f_i is a boolean function $f_i : S \longrightarrow \{0, 1\}$, to which we will refer from now on as a *Rule*, $c \leq c_{max}$, $c \in \mathbb{N}$, is the *DC* size and $c_{max} \in \mathbb{N}$ is an upper bound for the DC size. Motivated by Sect. 2.2 a *Rule* must be a boolean function in a *Conjunctive Normal Form* consisting of at most two single-literal clauses. An example of a *DC* is presented below.

$$\{miR\text{-}a,\ miR\text{-}b \wedge \neg miR\text{-}c,\ miR\text{-}a \wedge miR\text{-}d\}.$$

We assume that each *Rule* in the set must be unique, i.e., we do not allow copies of *Rules* in the *DC*. Also, two identical miRNA IDs cannot occur in one *Rule*, i.e., a trivial false function is not allowed ($a \wedge \neg a$). Thus, the functions are in a minimised form ($a \wedge a = a$). The DC categorizes cells according to a threshold function $F_{DC} : S \longrightarrow \{0, 1\}$ with

$$F_{DC}(s) = \begin{cases} 0, & \sum_{i=1}^{c} f_i(s) < \theta \\ 1, & \sum_{i=1}^{c} f_i(s) \geq \theta, \end{cases} \tag{2}$$

where $s \in S$ is a sample and $\theta \in [0, c]$ is a threshold. Here, we use $\theta = \lfloor \alpha \cdot c \rceil$ as the threshold, where α is a ratio that allows to calculate the decision threshold based on the classifier size. The threshold is then rounded half up. F_{DC} returns 1/True if a certain number of *Rules* (θ) outputs 1/True, e.g., $\alpha = 0.5$ for c =5 indicates that at least 3 *Rules* must output 1/True to classify a cell as positive.

Depending on α one may receive different results. In case of a very low threshold, e.g., if only one *Rule* outputing 1/True results in *DC* outputing 1/True, the DC becomes simply a disjunction of *Rules*. Note, that the function may then classify in favor of the positive class, as the decision to classify a sample as positive is in fact made by only one rule. This effect is already reduced by not considering 2-literal OR gates as rules. Otherwise, if the threshold is c ($\alpha = 1$), i.e., all the rules must output 1 for the DC to output 1, the function takes a form of a conjunction of clauses staying close to the single-circuit classifier. Unlike the disjunction, a conjunction may classify in favor of the negative class which may decrease the sensitivity of the method. Applying intermediate thresholds results in different combinations of those functions, therefore, different classification performance. In terms of cell classifiers applied as a cancer treatment, one may consider the following problem: in case of high α, the classifier may misclassify the diseased cells resulting in false negatives. Thus, the treatment may be less effective. However, low α may result in misclassification of healthy cells which

makes the treatment more toxic as the drug is released in those cells (false positives). Here, one should consider what type of errors is less desirable and apply a suitable threshold. We discuss this issue further in Sect. 4.

2.4 Evaluation

Here, we introduce the measures we employ to evaluate DCs in terms of binary classification. Many metrics that may be applied are available [20]. However, real-world expression data is often heavily imbalanced, i.e., the samples are not equally represented in the two classes. Data imbalance may significantly influence the classification results [27]. Balanced Accuracy (BACC) is an intuitive and easily interpretable metric that allows to balance the importance of samples in both classes (Eq. 3) [20]. Thus, as a main measure of classifier's performance we apply BACC.

$$BACC(DC, D) = \frac{\frac{TP}{P} + \frac{TN}{N}}{2} \tag{3}$$

where D is a given data set, P and N are the numbers of positive and negative samples in D, TP is the number of samples correctly classified as positive and TN is the number of samples correctly classified as negative. TP and TN are threshold-dependent values, i.e., they may change while applying different threshold values for a given classifier. To evaluate other aspects of classifier's performance we employ additional common metrics such as sensitivity ($TP/(TP + FN)$), specificity ($TN/(TN + FP)$) and accuracy ($(TP + TN)/(P + N)$). Sensitivity represents the ability of the method to correctly distinguish samples belonging to the positive class, while specificity shows the ability to correctly distinguish those belonging to the negative class. Accuracy gives information about the proximity of results to the true values, but does not take data imbalance into account.

3 Genetic Algorithm

In this section we present the architecture of a GA applied to design and optimization of DCs. In the following sections we describe the core structure of the algorithm as well as the used parameters and operators.

3.1 General Description

The input miRNA expression data must be formatted as described in Sect. 2.1. To optimize the DCs, seven parameters must be specified: *iter* - number of GA's iterations, *ps* - population size, i.e., the number of DCs allowed in the population, *cp* - crossover probability, *mp* - mutation probability, *ts* - tournament size, c_{max} - maximal size of a classifier, i.e., the number of single-circuit classifiers in a DC, α - the decision threshold ratio. As an output, the algorithm returns a list of all best solutions found over the GA's iterations according to their balanced accuracy (DC_{best}). In case of single-circuit classifiers, besides the accuracy, the

complexity of a solution is also taken into account [1,15]. Thus, we choose the solution consisting of the lowest number of rules as the optimal one. The algorithm starts with a random generation of an initial population (Algorithm 1, line 1). Next, the population is evaluated and a list of best solutions DC_{best} is created (Algorithm 1, 2). Having the initial population generated, the algorithm starts with a first generation. At the beginning, *ps* individuals are selected in so-called tournaments as potential parents to be recombined, i.e., randomly exchange genes. (Algorithm 1, 4–7). Many selection operators are described in the literature. Tournament selection allows to increase the chance of very good solutions to be selected as parents while maintaining the diversity in the population and can be efficiently implemented [23]. Next, the crossover occurs with the probability *cp* (Algorithm 1, 8–13). Crossover allows to generate new solutions (children), based on previously selected individuals (parents). Here, a child classifier may be created by copying rules from parent classifiers by randomly choosing which parent the next rule is duplicated from. As classifier sizes may differ, we propose two recombination strategies described further in Sect. 3.4. Next, individuals in the new population may mutate with the probability *mp* (Algorithm 1, 14). At the end of each iteration the list of best solutions (DC_{best}) is updated (Algorithm 1, 15). All the described steps in a generation are repeated *iter* times (Algorithm 1, 3–16). Below we explain the details of the algorithm design.

Algorithm 1. A genetic algorithm for designing DCs.

Data: dataset D
Parameters : number of iterations *iter*, population size *ps*, crossover probability *cp*, mutation probability *mp*, tournament size *ts*, maximal size of DC c_{max}, threshold ratio α
Output: DC_{best}

```
1  Population ⟵ InitializePopulation(D, ps, c_max)
2  DC_best ⟵ Evaluate(Population, D, α)
3  for i = 0 to iter do
4      for i = 0 to ps/2 do
5          Parent_1, Parent_2 ⟵ SelectParents(Population)
6          Parents ⟵ Add(Parent_1, Parent_2)
7      end
8      for i = 0 to ps/2 do
9          Parent_1, Parent_2 ⟵ RandomlyChooseParents(Parents)
10         Child_1, Child_2 ⟵ Crossover(Parent_1, Parent_2, cp, c_max)
11         NewPopulation ⟵ Add(Child_1, Child_2)
12         RemoveUsedParents(Parent_1, Parent_2, Parents)
13     end
14     Population ⟵ Mutate(NewPopulation, D, mp, c_max)
15     DC_best ⟵ Evaluate(Population, D, α)
16 end
```

3.2 Population

Individual Encoding. An individual (i.e., a DC) is encoded as a vector of single rules (genes). A unique ID and a fitness score is assigned to each individual. Both, the distributed classifier and single rules must satisfy the previously described constraints (see Sect. 2.3). Note, rules must consist of unique miRNA-inputs and DCs must consist of unique rules.

Initial Population. An initial population of a given size (ps) is generated randomly, i.e., each classifier and each single rule in the classifier is randomly initialized. Individuals in the population may be of a different size c and maximally of a size c_{max}. Thus, to generate a new individual, c must first be defined. Then, each single rule is generated in a few steps. First, the rule size ($RuleSize$) and $RuleSize$ miRNA IDs are randomly chosen. Then, for each miRNA the sign (positive/negative) is randomly assigned. This procedure (Algorithm 1, 1) is described in details in the appendix (Algorithm 2).

3.3 Fitness Function and Evaluation

As described in Sect. 2.4, to evaluate the classification performance of a distributed classifier we apply balanced accuracy as the fitness function. To count TPs and TNs we iterate over samples and evaluate the performance of a DC according to the threshold function described in Sect. 2.3. The fitness score is calculated separately for each DC in the population (Algorithm 1, 2, 15). As mentioned before, each iteration of the GA is completed by the update of the list of the best found solutions (DC_{best}). If the newly generated DCs perform with higher BACC than the solutions currently stored in DC_{best}, the new best DCs replace the previously found classifiers. If the new DCs have identical scores as the solutions in DC_{best} they are added to the list of the best solutions (Algorithm 1, 15). The classification threshold is a parameter specified by the user. In Sect. 4 we discuss the influence of different thresholds on the results.

3.4 Operators

Selection. Parents, to be potential candidates for recombination, are chosen in a process of *tournament selection* (Algorithm 1, 4–7). In each selection iteration two parents are chosen in separate tournaments. To select one parent, a number of ts individuals is randomly chosen from the current population to participate in a tournament. The winning candidate is an individual with the best fitness score. The first and the second parent must be different individuals. Thus, in each iteration, after choosing the first parent, its ID is temporarily blocked to be re-selected. The steps are repeated to form a population of selected individuals of the size ps. For more details see Appendix (Algorithm 3).

Crossover. In each crossover iteration two parents are randomly chosen from a population of selected individuals to recombine and generate two new individuals. Crossover (Algorithm 1, 8–13) occurs with the probability *cp*. To decide whether parents exchange information a random number p is chosen. If $p \leq cp$ then the two randomly chosen parents recombine. Otherwise, parents are copied to a new population. If chosen parents are of the same size we perform uniform crossover (Fig. 2A). To create two new individuals, rules from the first and second parent are paired off. Then, the first rule in each pair is assigned with equal probability to either the first or second child, while the second rule is assigned to the other child. The step is repeated until all the rules from parents were utilized and the children consist of the same number of rules as the parents. Otherwise, if the sizes of parents differ, to preserve a chance for each rule to be exchanged, we apply an index-based crossover (Fig. 2B). Here, the rules from the first and second parent are paired off according to a randomly chosen index specifying the position of a shorter parent in relation to the other one (see example in Fig. 2B). Paired rules are crossovered uniformly. Rules that cannot be paired (due to different sizes) may be copied to a randomly chosen child. Note, the index-based crossover may shorten the size of an individual as additional rules cannot be copied to the larger classifier. Details on the implementation of the index-based crossover may be found in the Appendix (Algorithms 4 and 5).

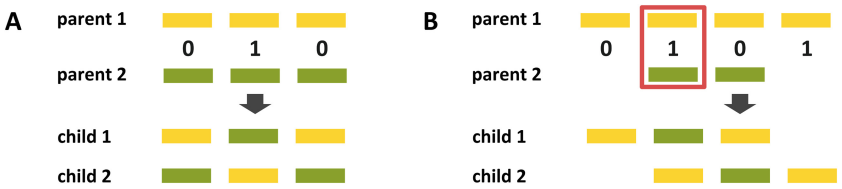

Fig. 2. Two crossover strategies applied in the presented GA. Yellow (light) and green (dark) boxes represent rules in different DCs (parents or children). (A) Uniform crossover. (B) Index-based crossover. The crossover index is marked by a red frame. (Color figure online)

Mutation. Mutation (Algorithm 1, 14) may occur on two levels: both, rules and inputs may mutate. A rule may (i) be removed from a classifier, (ii) be added to a classifier and (iii) be copied from one classifier to another. As mentioned before, index-based crossover may shorten the classifier. Here, two possibilities to extend the size of a classifier are available: a new rule may be initialized and added to a classifier or copied from another classifier. These two options balance the influence of crossover on the size of classifiers. An input may (i) be removed from a rule, (ii) be added to a rule, (iii) may change the sign (i.e., become a negative or positive input respecting the constraints described in Sect. 2.3). Rules, being larger components affecting the classifier size, mutate with a lower probability than inputs (0.2). Note, the maximal size of a classifier (c_{max}) must be preserved. For more details see Appendix (Algorithm 6).

4 Case Study

In this section, we illustrate the potential of DCs in application by performing case studies on real-world breast cancer data. We first describe the data sets used to evaluate DC performance. Then, we present results of parameter tuning and cross-validation. We analyze the classifier performances, as well as the relevance of chosen miRNAs. Finally, we compare DCs with a single-circuit classifier design approach.

4.1 Breast Cancer Data

To evaluate the performance of our approach we use Breast Cancer data sets previously applied by Becker et al. [1] and Mohammadi et al. [15] to the design of single-circuit classifiers. Originally the data was described by Farazi et al. [5] and pre-processed by Mohammadi et al. [15]. The details about the samples and miRNAs may be found in Table 2. The data set *All* includes samples of different breast cancer subtypes. This allows to compare breast cancer samples with the control samples. The following data sets are subsets representing different breast cancer subtypes containing information about the differences between particular subtypes and the control. Note, the data sets are significantly imbalanced as the negative class is heavily underrepresented. The data is formatted according to the description presented in Sect. 2.1. In terms of cell classifiers, non-regulated miRNAs do not carry any information. Thus, we remove them from the data sets before optimizing the classifiers. The last two columns of Table 2 include numbers of miRNAs before and after the filtering procedure.

Table 2. Breast Cancer data description.

Dataset	Samples	Positive	Negative	miRNAs	filtered miRNAs
All	178	167	11	478	57
Triple-	82	71	11	456	52
Her2+	86	75	11	438	19
ER+ Her-	32	21	11	392	18
Cell Line	17	6	11	375	59

4.2 Parameter Tuning

To tune the parameters of the genetic algorithm we applied a random search approach. The random search method allows to obtain results similar to the grid search approach, while decreasing the computational cost [2]. This provides an opportunity to extend the range of tested parameters. To tune the parameters we used the Breast Cancer All data set. We performed 3-fold cross-validation and repeated each GA run 10 times to obtain the average balanced accuracy on

the test data. We have randomly chosen 300 combinations of 5 parameters in following ranges: *iter*: 25–100, step 25; *ps*: 50–300, step 50; *cp*: 0.1–1.0, step 0.1; *mp*: 0.1–1.0, step 0.1; *ts*: 10–50%, step 10% (of *ps*). We tuned the parameters for $\alpha = 0.50$ as in intermediate threshold ratio and $c_{max} = 5$ and chose a following set of parameters based on average scores: $iter = 75$, $ps = 200$, $cp = 1.0$, $mp = 0.3$, $ts = 10\%$ (20 individuals). We applied those parameters to all case studies presented in the following sections. One may expect that the parameters optimized for a given decision threshold may further improve the performance of classifiers. We comment on it briefly in Sect. 5.

4.3 Cross-Validation

To evaluate the classifiers accuracy we performed 3-fold cross-validation for the breast cancer data sets presented in Sect. 4.1. We partition the data sets into 3 folds nearly equal in terms of the number of samples representing each class per fold. For each fold we run the algorithm once. For all tests we apply $c_{max} = 5$. The classifier size $c_{max} = 5$ allows to preserve the maximal number of miRNA inputs as proposed for single-circuit classifiers [1,15]. Maintaining similar complexity of classifiers allows to compare the DC-based method to another approach.

We test eight different values of α: 0.25, 0.35, 0.40, 0.50, 0.60, 0.65, 0.75, 0.85, to evaluate the influence of the threshold function on the classification accuracy. As mentioned before, the results might be influenced by the parameter tuning being done for $\alpha = 0.5$. The best results are presented in Table 3 (complete results for different α values may be found in the Appendix, Table 5). The DCs presented in the results are the first best shortest classifiers found by the algorithm. If identical BACC values for the testing data were obtained for more than one α, we present results for a DC with the highest BACC value on the training data. In case of equal training BACC values, we present an exemplary result for a chosen threshold. Table 3 includes the α-s and performance scores. All scores except of BACC_{train} were calculated on the testing data.

Table 3. Results of 3-fold cross-validation. For the Breast Cancer All data set we found DCs performing with identical score values for two α values (0.50, 0.60) and for ER+Her- for 6 different α values (0.35, 0.50, 0.60, 0.65, 0.75, 0.85)

Dataset	α	Sensitivity	Specificity	ACC	BACC	BACC_{train}
All	0.50	0.92	0.92	0.92	0.92	0.98
Triple-	0.85	0.92	0.75	0.89	0.83	0.98
Her2+	0.75	0.99	0.61	0.94	0.80	0.96
ER+ Her-	0.50	0.90	0.64	0.82	0.77	0.93
Cell Line	0.25	1.00	1.00	1.00	1.00	1.00

High BACC values obtained for the training data sets, as well as the average final population BACC values (0.91), show that the populations converge over the iterations resulting in high-performing DCs. The BACC values measured for the testing data sets are significantly higher for the largest and the smallest data sets than for the intermediate-size ones. Note, that for the data sets Her2+ and ER+ Her- the number of relevant miRNAs is significantly lower than for the other data sets. Thus, the space of available solutions is also substantially decreased. Cell Line data set includes 6 different miRNAs that perfectly separate samples [1]. Thus, excellent performance was expected for this particular data set. The accuracy is higher than BACC for all data sets as the metric is not sensitive to data imbalance.

The sensitivity is high for all data sets meaning that the method successfully classifies samples belonging to a positive class. However, the specificity is substantially decreased for Her2+ and ER+ Her- data sets. Note, the data sets are significantly imbalanced, i.e., the negative class is strongly underrepresented. Thus, even small number of errors results in substantially decreased specificity.

The best α values differ among the data sets. For the largest one, α is equal or not much higher than 50%. The data sets of intermediate sizes (Triple- and Her2+) favoured two more extreme α values. For the ER+Her- several α values returned identical results (Appendix, Table 5). For the smallest data set the lowest α value resulted in the highest BACC. Thus, the threshold seems to be data-specific and should be adjusted to the data set for the DC to perform well.

Applying a certain threshold caused a shift in the rates of certain types of errors. Here, we analyze false positive rates (FP_{rate}) and false negative rates (FN_{rate}) observed among all data sets for two extreme applied α values. In case of a low threshold (0.25, $FP_{rate} = 0.34$) the shift is displayed towards misclassification of the negative samples in comparison to a very high threshold (0.85, $FP_{rate} = 0.27$). The high threshold (0.85, $FN_{rate} = 0.13$) causes more frequent misclassification of positive samples in comparison to the low one (0.25, $FN_{rate} = 0.04$). The influence of a certain threshold on the shift should be further investigated. Complete information about FP_{rate}s and FN_{rate}s for different thresholds may be found in Appendix, Table 6.

The tests were performed using Allegro CPU Cluster provided by Freie Universitaet Berlin[1]. An average run-time is 45 min for one cross-validation fold of the largest data set employed in the case studies. The tests may be performed on a personal computer. However, the breast cancer data sets consist of up to 180 samples and up to 60 relevant miRNAs. Thus, one should consider performing extended scalability tests to estimate the run-time limits of the method.

4.4 Analysis of Input Viability

In this section we analyze miRNA inputs that occur in two exemplary classifiers. We chose the best performing classifiers for the largest data set (All) representing all subtypes and the smallest Cell Line data set.

[1] https://www.allegro.imp.fu-berlin.de/Cluster.

For breast cancer All two different α values resulted in the highest BACC. We found that classifiers for each cross-validation fold in the data set are identical for both α values. Also, all the classifiers are of the same size $c = 4$. In this case the applied α does not change the threshold function between both values (0.50 and 0.60), i.e., for all data sets at least 2 *Rules* must output 1 to classify a cell as positive. Below we present a DC found for the third cross-validation fold of the All data set. The classifier consists of 4 different 1-input rules. We analyzed the miRNAs and found that all of them may be relevant for cancer sample classification. The classifier is presented below.

$$\{\neg miR\text{-}378,\ miR\text{-}200c,\ \neg miR\text{-}145,\ \neg miR\text{-}451\text{-}DICER1\}$$

miR-378, miR-145, miR-451-$DICER1$ are described as down-regulated in breast cancer [4,5], e.g., the study by Ding et al. [4] has shown that underexpression of miR-145 is related to increased proliferation of breast cancer cells. Also, miR-378 occurred as down-regulated in the best 1-input single-circuit classifier presented previously by Becker et al. for the same data set [1]. miR-200c is marked as up-regulated in breast cancer in [21].

Another classifier we present is a DC for the third cross-validation fold for the Cell Line data set:

$$\{\neg miR\text{-}146a,\ \neg miR\text{-}143\}$$

For most of the α values the performance of found DCs was very low for this particular fold in the Cell Line data set (BACC = 0.50). A perfect classifier of size 2 performing with BACC = 1.00 on both training and testing data was found with $\alpha = 0.25$, i.e., one of 2 rules must output 1 to classify the cell as positive. We found that both, miR-146a and miR-143, are described as down-regulated in breast cancer [10,16].

4.5 Comparison to Other Methods

We optimized single-circuit classifiers with the ASP-based method proposed by Becker et al. [1] by performing 3-fold cross-validation using the same data sets and identical division into folds. The objective function of the ASP algorithm is based on the minimization of the total number of classification errors. Note that the ASP method may return several optimal classifiers. Different combinations of FPs and FNs influence the balanced accuracy. Thus, to increase the chance of ASP to perform well, we have chosen the best classifiers according to their BACC. Here, we do not compare our results to Mohammadi et al. [15] as the approach did not perform better than the ASP-based approach as described by Becker et al. in terms of binary classification [1] (Table 4).

Table 4. Comparison of results of 3-fold cross-validation for the ASP-based approach proposed by Becker et al. [1] and for the GA (as in Table 3).

Dataset	Method	Sensitivity	Specificity	ACC	BACC	$BACC_{train}$
All	GA	0.92	0.92	0.92	**0.92**	0.98
	ASP	0.96	0.47	0.93	0.72	0.92
Triple-	GA	0.92	0.75	0.89	**0.83**	0.98
	ASP	0.89	0.44	0.83	0.67	0.96
Her2+	GA	0.99	0.61	0.94	0.80	0.96
	ASP	1.00	0.61	0.95	**0.81**	0.96
ER+ Her-	GA	0.90	0.64	0.82	**0.77**	0.93
	ASP	0.90	0.64	0.82	**0.77**	0.93
Cell Line	GA	1.00	1.00	1.00	**1.00**	1.00
	ASP	0.83	1.00	0.93	0.92	1.00

The DC-based method outperformed the single-circuit approach in 3 of 5 case studies. For two other data sets the resulting BACC (test) values are either identical (ER+Her-) or very similar (Her2+). This may imply that further improvement of classifier performance for those data sets is not possible with the currently applied techniques. The training BACC values are also significantly higher for the DC-based approach. Note, the DC-based design method explores a different search space than the single circuit approach. Although single circuits are also allowed as 1-rule classifiers, their complexity is substantially lower in comparison to single circuits. Additionally, ASP returns globally optimal solutions, i.e., it adjusts the classifier perfectly to the training data, which may cause overfitting. Although, the classifiers obtain high BACC on the training data (average for all data sets: 0.95), the classifiers may be too specific to perform well on the testing data.

The scalability of the ASP-based approach was previously shown by Becker et al. [1]. As mentioned before, the ASP-based approach optimizes much simpler classifiers than the GA-based method. Thus, the run-times of both approaches are not easily comparable.

5 Discussion

In this article, we introduced a new approach to cell classifier design. The concept of DCs proposed by Didovyk et al. [3] was re-formalized in the context of miRNA-based cell classification. We designed and implemented a genetic algorithm that allows design and optimization of *DC*s. We performed case studies on real-world data and compared our results to a single-circuit design method obtaining significantly higher or similar accuracy.

DCs show immense potential as an alternative to single-circuit designs. Presented case studies demonstrate the DC's ability to perform classification on real-world cancer data. The results obtained on the training data show that the proposed genetic algorithm allows to optimize classifiers that achieve high accuracy. The cross-validation demonstrates that the optimized DCs classify unknown data with high accuracy. The data sets for which the algorithm returns the worst results (Her2+, ER+Her-) are ones with the lowest number of relevant miRNAs. Thus, the number of possible solutions is significantly decreased in contrast to other data sets. The best performing decision thresholds differ significantly among data sets. However, higher α values seem to be more efficient. Testing a wide range of thresholds while optimizing the classifiers is strongly recommended. The comparison to a single-circuit design method shows that DCs outperformed single-circuit classifiers on most of the presented data sets according to balanced accuracy. Although the GA performs better on the largest and the smallest data sets than on the intermediate-size ones, the results obtained for both compared methods for Her2+ and ER+Her- are very similar which may suggest that for those data sets significant improvement is not possible. The improvements in binary classification may be a result of applying a different strategy to cell classifier design. Here, single-circuit decision is complemented by a collective classification based on a threshold function. Thus, the DCs may be more resistant to data noise than single-circuit classifiers.

Generally, the problem of designing reliable and efficient DCs begins with the initial data processing. As mentioned before, the data sets employed for the case studies are significantly imbalanced. Although we apply an objective function that allows to partially overcome this issue, one may consider applying data balancing methods such as weighted schemes that balance the sample importance [7]. Furthermore, our approach to the design of DCs is based on binarized data sets. As mentioned before, data discretization allows obtaining clear-cut information about miRNA regulation and efficient exploration of the search space. One advantage of this data processing procedure is absorption of noise coming from, e.g., lab artifacts. However, simultaneously some information that may be valuable for the classification is lost. Considering binarization according to a given threshold, miRNAs having their concentrations significantly higher (or lower, respectively) than the threshold may be more informative. Thus, one may introduce a multi-objective function that allows to optimize both, the accuracy and the use of particular miRNAs according to, e.g., a weighted scheme favoring more reliable miRNAs.

Adapting the ASP approach to classifier design, one could apply ASP to the optimization of DCs, obtain globally optimal solutions and compare with the heuristic approach. However, ASP searches through the entire solution space; thus, the run-time may be significantly increased with the rising number of possible combinations. As we expect that this may significantly limit the ASP-based optimization, one may explore other possibilities. In the proposed GA the initial population is generated randomly, i.e., there is no preference in choosing

particular miRNAs or gate signs that built rules. One may optimize the initial population by creating rules taking such preferences into account. The ASP allows to optimize single short classifiers with relaxed constraints in a short time, e.g., allowing up to a certain number of errors. This may generate a pool of rules that are pre-optimized resulting in a better starting point for the algorithm.

Although the results demonstrate that classifiers perform with high accuracy, the possibilities to further develop the presented method should be explored. Certainly, the approach must be tested using more data representing variety of cancer types. Although the proposed genetic algorithm performed well on the presented case studies, particular parts of the algorithm may be improved. In this work we do not tune parameters for different applied thresholds due to time-consuming calculations. It should be further investigated whether the parameters may be optimized for certain thresholds to improve the performance of classifiers. Additionally, different selection operators may be tested to evaluate the influence of a chosen operator on the results [23]. Although tournament selection is described in the literature as a well-performing operator, some other operators may be more accurate for particular problems than the commonly recommended ones.

Although DCs are not yet applied in terms of cancer cell classification, the approach should be further investigated. DCs are designed based on available building blocks that are in fact single-circuit classifiers. Mohammadi et al. [15] presented a biochemical model of a single-circuit classifier that allows to manipulate the output compound concentration. Thus, the biological output threshold for a given classifier may be adjusted to perform the classification in living cells. As the on-off single-circuit response may be regulated on the biological level, the sum of their outputs should also be adaptable for a given DC. This needs to be investigated through further work in the lab.

Data and Software Availability. The algorithm is implemented in Python 3. The scripts, as well as the data used to tune the parameters and test the algorithm's performance including the results, are available at GitHub [18].

Aknowledgements. We would like to thank to P. Mohammadi, Y. Benenson and N. Beerenwinkel (ETH Zurich) for sharing the breast cancer data with us. MN would like to thank to J. Bartoszewicz (RKI, Berlin) for his valuable comments and support with cluster handling.

Appendix

Algorithm 2. The algorithm describes the generation of an initial population of size *ps* (Sect. 3.2). *ps* individuals are created randomly, i.e., the number of rules is randomly chosen and the rules are randomly generated. To create an individual its size must be first specified (line 2). Then, *c* rules must be generated in a

few steps. First, the size of a rule ($RuleSize$) and miRNA IDs ($miRNAs$) must be randomly chosen (lines 4–7). Then, the sign (positive/negative) is randomly assigned to the miRNAs (line 8). Note that in case of $RuleSize = 2$, the miRNAs are connected with an AND. c rules generated as described above create an individual which may be added to a population (line 12). The steps are repeated until the population consists od ps individuals (lines 1–11).

Algorithm 2. Initialization of a first population.

Data: dataset D
Parameters: population size ps, maximal size of a DC c_{max}
Output: *Population*

```
1  for i = 1 to ps do
       /* randomly choose the size of a new classifier */
2      c ⟵ RandomlyChooseInRange(1, c_max)
3      for i = 1 to c do
           /* randomly choose the size of a new rule */
4          RuleSize ⟵ RandomlyChooseInRange(1, 2)
           /* randomly choose miRNA IDs */
5          miRNAs ⟵ RandomlyChooseIDs(D, RuleSize)
           /* randomly assign miRNA signs */
6          miRNAs ⟵ RandomlyAssignSigns()
           /* create a new rule */
7          Rule ⟵ CreateARule(miRNAs)
           /* add a new rule to a classifier */
8          Individual ⟵ Add(Rule)
9      end
       /* add a new classifier to a population */
10     Population ⟵ Add(Individual)
11 end
```

Algorithm 3. The algorithm describes the selection of parents that are potential candidates to recombine (Sect. 3.4). The parents are chosen in tournaments of size ts, i.e., ts candidates are randomly chosen from the population to participate in a tournament (lines 1–5, 7–11). In each round 2 parents are selected from the population. The winning candidates are individuals with the highest BACC (lines 5, 11). After the first parent is selected its ID is temporarily blocked to be re-selected (line 6). This allows to diverse the population of selected parents. The new population of selected parents is then utilized to perform crossover.

Algorithm 3. Selection of parents

Input: *Population*
Parameters: population size ps, tournament size ts
Output: $Parent_1$, $Parent_2$

```
   /* repeat adding to a tournament ts times */
1  for i = 1 to ts do
       /* randomly choose an individual's ID */
2      Candidate ⟵ RandomlyChooseInRange(1, ps)
       /* add a candidate ID to a tournament */
3      Candidates ⟵ Add(Candidate)
4  end
   /* choose the best parent in a tournament */
5  Parent_1 ⟵ SelectBest(Candidates)
   /* block a chosen ID to be re-selected */
6  ps ⟵ BlockID(Parent_1, ps)
7  for i = 0 to ts do
       /* randomly choose an individual's ID */
8      Candidate ⟵ RandomlyChooseInRange(1, ps)
       /* add a candidate ID to a tournament */
9      Candidates ⟵ Add(Candidate)
10 end
   /* choose the best parent in a tournament */
11 Parent_2 ⟵ SelectBest(Candidates)
```

Algorithm 4. The algorithm describes the crossover procedure performed on the population of selected parents (Sect. 3.4). Each couple of parents chosen randomly from the population of selected parents exchange genes with the probability cp. If the randomly chosen *probability* is lower than cp the parents undergo the crossover (lines 2–13). Otherwise, the parents are copied directly to a new population (line 15). If parents are of the same size, uniform crossover is performed (line 11–12). Otherwise, index-based crossover is applied (lines 7–9). Both procedures are described in details in Sect. 3.4.

Algorithm 5. The algorithm describes the index-based crossover that we apply if the sizes of parents differ to preserve a chance for each rule to be exchanged. Here, the rules from the first and second parent are paired off according to a randomly chosen index specifying the position of a shorter parent in relation to the other one. The index is chosen randomly and is in range between 1 and $ParentSize_1$-$ParentSize_2$ (Algorithm 4, line 7). Paired rules are crossovered uniformly. Rules that cannot be paired (due to different sizes) may be copied to a randomly chosen child. As a result, the number of rules in each child is between the minimum and the maximum size of the two parents. Note, the index-based crossover may shorten the size of an individual as additional rules cannot be copied to the larger classifier. This procedure is described in details in Sect. 3.4.

Algorithm 4. Crossover

Input: $Parent_1$, $Parent_2$, crossover probability cp
Output: $Child_1$, $Child_2$

```
   /* randomly choose the probability of crossover */
1  probability ⟵ DrawProbability(0,1)
   /* if probability ≤ cp perform crossover */
2  if probability ≤ cp then
       /* assign a longer parent to Parent_1 */
3      Parent_1, Parent_2 ⟵ AssignParentsBySize(Parent_1, Parent_2)
       /* assign sizes of parents */
4      ParentSize_1 ⟵ Size(Parent_1)
5      ParentSize_2 ⟵ Size(Parent_2)
       /* if parents sizes differ perform index-based crossover */
6      if ParentSize_1 ≠ ParentSize_2 then
           /* randomly choose the crossover index */
7          CrossoverIndex ⟵ RandomlyChooseInRange(1, ParentSize_1 -
           ParentSize_2)
           /* perform index based crossover */
8          Child_1, Child_2 ⟵ IndexCrossover(Parent_1, Parent_2,
           ParentSize_1, ParentSize_2, CrossoverIndex)
9          Population ⟵ Add(Child_1, Child_2)
10     else
           /* if parents have identical sizes perform uniform crossover
              */
11         Child_1, Child_2 ⟵ UniformCrossover(Parent_1, Parent_2)
           /* add children to a new population */
12         Population ⟵ Add(Child_1, Child_2)
13     end
14 else
       /* if probability > cp copy parents to a new population */
15     Population ⟵ Add(Parent_1, Parent_2)
16 end
```

Algorithm 6. The algorithm describes mutation (Sect. 3.4). Mutation may occur on two levels: both, rules and inputs may mutate. A rule may (i) be removed from a classifier, (ii) be added to a classifier and (iii) be copied from one classifier to another (lines 5–17). An input may (i) be removed from a rule, (ii) be added to a rule, (iii) may change the sign i.e., become a negative or positive input (lines 19–32). Rules, being larger components affecting the classifier size, mutate with a lower probability than inputs (0.2). Note, the maximal size of a classifier (c_{max}) must be preserved.

Algorithm 5. Index-based crossover

Input: $Parent_1$, $Parent_2$, $ParentSize_1$, $ParentSize_2$, $CrossoverIndex$
Output: $Child_1$, $Child_2$

for $i = 1$ **to** $ParentSize_1$ **do**
  /* decide whether the rule will be exchanged */
  /* $SwapMask$=1 corresponds to rule exchange */
  /* $SwapMask$=0 corresponds to copying without exchanging */
  $SwapMask \longleftarrow$ RandomlyChooseInRange(0, 1)
  /* 1 - rule is exchanged */
  if $SwapMask = 1$ **then**
    /* if the rules do not pair off */
    /* i.e., there is no possibility to exchange rules */
    if $i < CrossoverIndex$ *OR* $i \geq CrossoverIndex + ParentSize_2$ **then**
      /* copy a rule from $Parent_1$ to $Child_2$ */
      $Child_2 \longleftarrow$ CopyRule($Parent_1$, i)
    else
      /* if the rules pair off exchange rules */
      /* copy a rule from $Parent_2$ to $Child_1$ */
      $Child_1 \longleftarrow$ CopyRule($Parent_2$, i)
      /* copy a rule from $Parent_1$ to $Child_2$ */
      $Child_2 \longleftarrow$ CopyRule($Parent_1$, i)
    end
  else
    /* 0 - rule is not exchanged */
    if $i < CrossoverIndex$ *OR* $i \geq CrossoverIndex + ParentSize_2$ **then**
      /* copy a rule from $Parent_1$ to $Child_1$ */
      $Child_1 \longleftarrow$ CopyRule($Parent_1$, i)
    else
      /* else copy rules to the parents without exchanging */
      /* copy a rule from $Parent_1$ to $Child_1$ */
      $Child_1 \longleftarrow$ CopyRule($Parent_1$, i)
      /* copy a rule from $Parent_2$ to $Child_2$ */
      $Child_2 \longleftarrow$ CopyRule($Parent_2$, i)
    end
  end
end

Algorithm 6. Mutation

```
   Input: Population, maximal size of a DC c_max
   Output: Population
1  for i = 1 to ps do
       /* randomly choose the probability of mutation */
2      probability ⟵ DrawProbability(0,1)
       /* if probability ≤ mp perform mutation */
3      if probability ≤ mp then
           /* choose the mutation level */
           /* 1 corresponds to mutation of a rule */
           /* 2-4 corresponds to mutation of an input */
4          MutationLevel ⟵ RandomlyChooseInRange(1, 5)
5          if MutationLevel = 1 then
               /* choose the mutation type */
6              MutationType ⟵ DrawItem(add, remove, copy)
7              switch MutationType do
8                  case add
                       /* add rule */
9                      AddRule(Population, i, c_max)
10                 end
11                 case copy
                       /* copy rule */
12                     CopyRule(Population, i, c_max)
13                 end
14                 case remove
                       /* remove rule */
15                     RemoveRule(Population, i)
16                 end
17             endsw
18         else
               /* choose the mutation type */
19             MutationType ⟵ DrawItem(add, remove, sign)
20             switch MutationType do
21                 case add
                       /* add input */
22                     Rule ⟵ DrawRule(1, ps)
23                     AddInput(Population, i, Rule)
24                 end
25                 case remove
                       /* remove input */
26                     RemoveInput(Population, i, c_max, Rule)
27                 end
28                 case sign
                       /* change sign of an input */
29                     ChangeInputSign(Population, i, Rule)
30                 end
31             endsw
32         end
33     end
34 end
```

Table 5. Results of 3-fold cross-validation.

Dataset	α	Sensitivity	Specificity	ACC	BACC	BACC_{train}
All	0.85	0.89	0.83	0.88	0.86	0.96
	0.75	0.94	0.81	0.93	0.87	0.98
	0.65	0.95	0.72	0.93	0.83	0.98
	0.60	0.92	0.92	0.92	**0.92**	0.98
	0.50	0.92	0.92	0.92	**0.92**	0.98
	0.40	0.94	0.64	0.92	0.79	0.99
	0.35	0.97	0.72	0.96	0.85	0.99
	0.25	0.96	0.72	0.94	0.84	1.00
Triple-	0.85	0.92	0.75	0.89	**0.83**	0.98
	0.75	0.96	0.67	0.92	0.81	0.99
	0.65	0.94	0.58	0.89	0.76	1.00
	0.60	0.93	0.64	0.89	0.78	1.00
	0.50	0.93	0.64	0.89	0.78	1.00
	0.40	0.94	0.56	0.89	0.75	1.00
	0.35	0.94	0.53	0.89	0.74	0.99
	0.25	0.94	0.53	0.89	0.74	1.00
Her2+	0.85	0.99	0.44	0.92	0.72	0.96
	0.75	0.99	0.61	0.94	**0.80**	0.96
	0.65	0.99	0.53	0.93	0.76	0.96
	0.60	1.00	0.53	0.94	0.76	0.96
	0.50	1.00	0.53	0.94	0.76	0.96
	0.40	0.99	0.53	0.93	0.76	0.96
	0.35	1.00	0.53	0.94	0.76	0.96
	0.25	1.00	0.53	0.94	0.76	0.93
ER+ Her-	0.85	0.90	0.64	0.82	**0.77**	0.93
	0.75	0.90	0.64	0.82	**0.77**	0.93
	0.65	0.90	0.64	0.82	**0.77**	0.93
	0.60	0.90	0.64	0.82	**0.77**	0.93
	0.50	0.90	0.64	0.82	**0.77**	0.93
	0.40	0.90	0.53	0.78	0.72	0.93
	0.35	0.90	0.64	0.82	**0.77**	0.93
	0.25	0.90	0.53	0.78	0.72	0.91
Cell Line	0.85	0.67	1.00	0.87	0.83	1.00
	0.75	0.67	1.00	0.87	0.83	1.00
	0.65	0.67	1.00	0.87	0.83	1.00
	0.60	0.67	1.00	0.87	0.83	1.00
	0.50	0.67	1.00	0.87	0.83	1.00
	0.40	0.67	1.00	0.87	0.83	1.00
	0.35	1.00	0.89	0.93	0.94	1.00
	0.25	1.00	1.00	1.00	**1.00**	1.00

Table 6. FP_{rate} and FN_{rate} values for different thresholds (for all datasets).

Threshold	FP_{rate}	FN_{rate}
0.85	0.27	0.13
0.75	0.23	0.11
0.65	0.31	0.11
0.60	0.26	0.12
0.50	0.26	0.12
0.40	0.35	0.11
0.35	0.34	0.04
0.25	0.34	0.04

References

1. Becker, K., Klarner, H., Nowicka, M., Siebert, H.: Designing miRNA-based synthetic cell classifier circuits using answer set programming. Front. Bioeng. Biotechnol. **6**, 70 (2018). https://doi.org/10.3389/fbioe.2018.00070
2. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**, 281–305 (2012)
3. Didovyk, A., Kanakov, O.I., Ivanchenko, M.V., Hasty, J., Huerta, R., Tsimring, L.: Distributed classifier based on genetically engineered bacterial cell cultures. ACS Synth. Biol. **4**(1), 27–82 (2015). https://doi.org/10.1021/sb500235p
4. Ding, Y., et al.: miR-145 inhibits proliferation and migration of breast cancer cells by directly or indirectly regulating TGF-β1 expression. Int. J. Oncol. **50**(5), 1701–1710 (2017). https://doi.org/10.3892/ijo.2017.3945
5. Farazi, T.A., et al.: MicroRNA sequence and expression analysis in breast tumors by deep sequencing. Cancer Res. **71**(13), 4443–4453 (2011). https://doi.org/10.1158/0008-5472.CAN-11-0608
6. Gallo, C.A., Cecchini, R.L., Carballido, J.A., Micheletto, S., Ponzoni, I.: Discretization of gene expression data revised. Brief. Bioinform. **17**(5), 758–770 (2016). https://doi.org/10.1093/bib/bbv074
7. Haixiang, G., Yijing, L., Shang, J., Mingyun, G., Yuanyue, H., Bing, G.: Learning from class-imbalanced data: review of methods and applications. Expert Syst. Appl. **73**, 220–239 (2017). https://doi.org/10.1016/J.ESWA.2016.12.035
8. Iorio, M.V., Croce, C.M.: MicroRNA dysregulation in cancer: diagnostics, monitoring and therapeutics. A comprehensive review. EMBO Mol. Med. **4**(3), 143–59 (2012). https://doi.org/10.1002/emmm.201100209
9. Lan, H., Lu, H., Wang, X., Jin, H.: MicroRNAs as potential biomarkers in cancer: opportunities and challenges. BioMed Res. Int. **2015**, 125094 (2015). https://doi.org/10.1155/2015/125094
10. Li, Y., Xu, Y., Yu, C., Zuo, W.: Associations of miR-146a and miR-146b expression and breast cancer in very young women. Cancer Biomarkers **15**(6), 881–887 (2015). https://doi.org/10.3233/CBM-150532
11. Manning, T., Sleator, R.D., Walsh, P.: Naturally selecting solutions: the use of genetic algorithms in bioinformatics. Bioengineered **4**(5), 266–78 (2013). https://doi.org/10.4161/bioe.23041

12. Marchisio, M., Stelling, J.: Computational design of synthetic gene circuits with composable parts. Bioinformatics **24**(17), 1903–1910 (2008). https://doi.org/10.1093/bioinformatics/btn330
13. McCall, J.: Genetic algorithms for modelling and optimisation. J. Comput. Appl. Math. **184**(1), 205–222 (2005). https://doi.org/10.1016/J.CAM.2004.07.034
14. Miki, K., et al.: Efficient detection and purification of cell populations using synthetic MicroRNA switches. Cell Stem Cell **16**(6), 699–711 (2015). https://doi.org/10.1016/j.stem.2015.04.005
15. Mohammadi, P., Beerenwinkel, N., Benenson, Y.: Automated design of synthetic cell classifier circuits using a two-step optimization strategy. Cell Syst. **4**(2), 207–218.e14 (2017). https://doi.org/10.1016/j.cels.2017.01.003
16. Ng, E.K.O., Li, R., Shin, V.Y., Siu, J.M., Ma, E.S.K., Kwong, A.: MicroRNA-143is downregulated in breast cancer and regulates DNA methyltransferases 3A inbreast cancer cells. Tumor Biol. **35**(3), 2591–2598 (2014). https://doi.org/10.1007/s13277-013-1341-7
17. Nielsen, A.A., et al.: Genetic circuit design automation. Science **352**, 6281 (2016). https://doi.org/10.1126/science.aac7341
18. Nowicka, M.: A genetic algorithm to designing distributed cell classifier circuits (2019). https://github.com/MelaniaNowicka/RAccoon
19. Palmer, E.: The T-Cell antigen receptor: a logical response to an unknown ligand. J. Recept. Signal Transduct. **26**(5–6), 367–378 (2006). https://doi.org/10.1080/10799890600919094
20. Ramola, R., Jain, S., Radivojac, P.: Estimating classification accuracy in positive-unlabeled learning: characterization and correction strategies. In: Pacific Symposium on Biocomputing, vol. 24, pp. 124–135 (2019)
21. Sánchez-Cid, L., et al.: MicroRNA-200, associated with metastatic breast cancer, promotes traits of mammary luminal progenitor cells. Oncotarget **8**(48), 83384–83406 (2017). https://doi.org/10.18632/oncotarget.20698
22. Schapire, R.E.: The strength of weak learnability. Mach. Learn. **5**(2), 197–227 (1990). https://doi.org/10.1007/BF00116037
23. Shukla, A., Pandey, H.M., Mehrotra, D.: Comparative review of selection techniques in genetic algorithm. In: 2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE), pp. 515–519. IEEE, February 2015. https://doi.org/10.1109/ABLAZE.2015.7154916
24. Slomovic, S., Pardee, K., Collins, J.J.: Synthetic biology devices for in vitro and in vivo diagnostics. Proc. Natl. Acad. Sci. **112**(47), 14429–14435 (2015). https://doi.org/10.1073/pnas.1508521112
25. Smith, R.W., van Sluijs, B., Fleck, C.: Designing synthetic networks in silico: a generalised evolutionary algorithm approach. BMC Syst. Biol. **11**(1), 118 (2017). https://doi.org/10.1186/s12918-017-0499-9
26. Xie, Z., Wroblewska, L., Prochazka, L., Weiss, R., Benenson, Y.: Multi-input RNAi-based logic circuit for identification of specific cancer cells. Science **333**(6047), 1307–1311 (2011). https://doi.org/10.1126/science.1205527
27. Yang, K., Li, J., Gao, H.: The impact of sample imbalance on identifying differentially expressed genes. BMC Bioinform. **7**(Suppl. 4), S8 (2006). https://doi.org/10.1186/1471-2105-7-S4-S8
28. Ye, H., Fussenegger, M.: Synthetic therapeutic gene circuits in mammalian cells. FEBS Lett. **588**(15), 2537–2544 (2014). https://doi.org/10.1016/j.febslet.2014.05.003