An ILP Implementation for Predicting Drug Sensitivity in Cancer Cell Lines

Tomas Raising

A thesis presented for the degree of
Bachelor of Science

Algorithmic Bioinformatics
Heinrich Heine University Düsseldorf
Germany
21st July, 2022

Acknowledgments

I am grateful for Prof. Dr. Gunnar Klau for providing guidance and introducing the interesting field of bioinformatics to me. Next, I want to thank Prof. Dr. Martin Lercher for agreeing to be my second assessor. I want to extend my gratitude Kerstin Lenhof for providing assistence with MERIDA. I also thank my daily supervisor, Nguyen Khoa Tran, who supported and encouraged me.

Computational infrastructure and support were provided by the Center for Information and Media Technology at Heinrich Heine University Düsseldorf.

Abstract

Cancer therapeutics can be further improved upon by correctly predicting medication effectiveness and inferring sensitivity information. However, interpretability of the methods and results plays a major role alongside performance.

We reproduce the input data and analyze the performance of MERIDA, an integer linear formulation based on LOBICO. To generate the data, we use the same GDSC datasets and data generation workflow as outlined in the MERIDA manuscript. The unavailability of older datasets seems to be a major bottleneck in this workflow.

After constructing our input data with help from the authors of MERIDA, we train both LOBICO and MERIDA on a selection of drugs using a stratified 5-fold cross-validation. We evaluate the results regarding runtime and statistical performance using specificity and sensitivity as metrics. We consider different weight functions, model parameters and a priori knowledge with MERIDA, if available. Our results show that when MERIDA and LOBICO an a model with feature size 4, MERIDA is faster up to a factor of 1241.02 on average. However, MERIDA does not benefit as much as LOBICO from different weight functions. LOBICO can achieve an improved runtime by 6.96 times using a cubic weight function instead a linear function. Statistically, LOBICO shows a higher specificity overall while MERIDA has a higher sensitivity on average. MERIDA shows a higher sensitivity overall on larger models and a higher specificity on smaller models. Neither LOBICO nor MERIDA seem to show a difference regarding weight functions.

In conclusion, MERIDA seems convincing regarding performance and interpretability.

Contents

1 Introduction

A major goal in oncology research is to further improve medicine and develop personalized treatments. Typically, machine learning algorithms are employed to analyze gathered data and infer drug sensitivity on cell lines. These models perform well in predicting drug sensitivity, however, the results are usually hard to interpret. An easily explainable model is important for wide acceptance and pharmaceutical application [1].

Improving interpretability, Knijnenburg et al. [2] present an integer linear program formulation to infer and predict drug sensitivity from binarized input data called Logic Optimization for Binary Input to Continuous Output (LOBICO). Although information loss is unavoidable in the data binarization process, LOBICO retains accuracy through weighting the binarized drug responses to retain information about the effectiveness of a drug against a specific cell line. LOBICO constructs logic formulas in *disjunctive normal form* (DNF), allowing the synthesis of any boolean function. The prediction vector is calculated by disjunction of intermediate terms, which are obtained by conjunction of the input features and their negations. However, building larger models might be considered unpractical because of the high computation time required.

Based on that approach, Method for Rule Identification with multi-omics Data (MERIDA) (Lenhof et al.) [3] utilizes another integer linear program formulation to classify cancer cell lines into sensitive and resistant classes. In contrast to LOBICO, MERIDA restricts its logic formula to a specific type by assuming that a cell line is only considered sensitive towards a drug if sensitivity-associated and no resistance-associated features are present. This allows MERIDA to build two terms, one that is a disjunction of sensitivity-associated features and the other a disjunction of resistance-associated features. A conjunction of those terms then results in the prediction vector. Additionally, MERIDA enables the inclusion of *a priori* knowledge while constructing the input matrix and also preselect features in the integer linear program (ILP) formulation.

Our goal is the reproduction and critical analysis of the results Lenhof et al. present. In Section 2 we explain important concepts that are vital for understanding this thesis. In Section 3 we describe how we generated and processed the data used in our workflow. Then we show our results and evaluations of the conducted experiments in Section 4. After that, we critically discuss these findings in Section 5 regarding reproducibility and interpretability. Finally, we draw a conclusion by comparing our obtained results to Lenhof et al.

2 Preliminaries

In this section we will briefly describe some theoretical components which are important for understanding the thesis.

2.1 Integer Linear Programming

An ILP is a mathematical optimization program in which all or some variables are integers. An example in canonical form is the following expression:

$$\max c^T x \tag{1}$$

$$\textit{subject to } Ax \leq b \tag{2}$$

$$x \in \mathbb{N} \tag{3}$$

ILPs have an objective function (1) that are subject to linear constraints (2). All variables are restricted to integers and, in this example, $x \geq 0$ (3) applies. A feasible solution is acquired by optimizing the objective function while adhering to the linear constraints. The vectors c, b and the matrix A are integer inputs, and the ILP is solved for the vector x.

2.2 k-Fold Cross Validation

k-fold cross-validation is a widely used technique to evaluate models. The dataset is split evenly into k disjoint subsets with one subset serving as the test set while the other $k-2$ subsets are used to train the modelt. Afterwards, the trained model is evaluated on the test set utilizing evaluation metrics. This process is then repeated k times so that each fold served as a test set once. A stratified k-fold cross-validation is the same procedure with the addition that the original proportion of classes is retained in each fold.

2.3 Evaluation Metrics

We use specificity and sensitivity to evaluate the test sets. These are metrics describe the ratio between *true positives* (TP), *false positives* (FP), *true negatives* (TN) and *false negatives* (FN).

$$\text{specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

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

Specificity defines the rate of a true negative result being recognized as negative. Sensitivity refers to the rate of a truly positive outcome being correctly identified as positive.

3 Methods

In this section we will describe where and how we obtained the data as well as the methods used to process and integrate it into our Snakemake pipeline. Although there does exist a reference implementation[1], we present a Python implementation of MERIDA using Gurobi and a Snakemake pipeline. However, the original scripts from this repository are used to generate binarized input data.

3.1 Data and Preprocessing

We use the same data sets as outlined by Lenhof et al. [3] in their Supplementary Information S1, specifically the GDSC database [4]. The Genomics of Drug Sensitivity in Cancer (GDSC) project aimed at identifying genetic features that predict drug sensitivity in over a thousand human cancer cell lines by screening them with a wide range of anti-cancer therapeutics.

We require the cell line information and drug response (logarithmized IC50 values) data from the GDSC1 and GDSC2 database. Furthermore, we downloaded mutation and copy number variation (CNV) datasets as well as gene expression information from the official GDSC website. The GDSC2 dataset is a more recent screening utilizing better equipment and improved procedures compared to GDSC1. Therefore, we will employ the GDSC2 dataset if there is information on a drug in both databases. We use additional data from CIViC [5], COSMIC [6], OncoKB [7] and the Cancer Genome Interpreter [8] to enrich the GDSC datasets and filter out relevant oncology and sensitivity information. As for genomic features, we only consider genes that are part of IntOGen [9] and a custom cancer gene driver list by Sanchez-Vega et al. [10]

Using the script collection provided by Lenhof et al. on their Github page, we generate a $N \times P$ matrix with each of the N cell lines containing information on P features. As the matrix is binarized in the process, the cell values indicate whether a feature is present ($= 1$) or absent ($= 0$) in a cell line. For each drug, we obtain one data matrix.

Unfortunately, we were unable to reproduce the binarized input data on our own since we could not acquire the specific GDSC and COSMIC database versions used by Lenhof et al. [3]. Also, we had difficulties operating the workflow on a step-by-step basis as the Python script documentation was rather scarce. However, after contacting the MERIDA authors, Lenhof provided us with the already processed data required and assisted us with our data reproduction efforts.

As a final step, we binarize the drug response by utilizing a drug-specific threshold on the GDSC IC50 values to classify all cell lines into the categories *sensitive* and *resistant*. We use the binarization thresholds provided by Lenhof et al., which were calculated with the procedure described by Knijnenburg et al. We then append the binarized drug response vector to the data

[1] *https://github.com/unisb-bioinf/MERIDA.git*

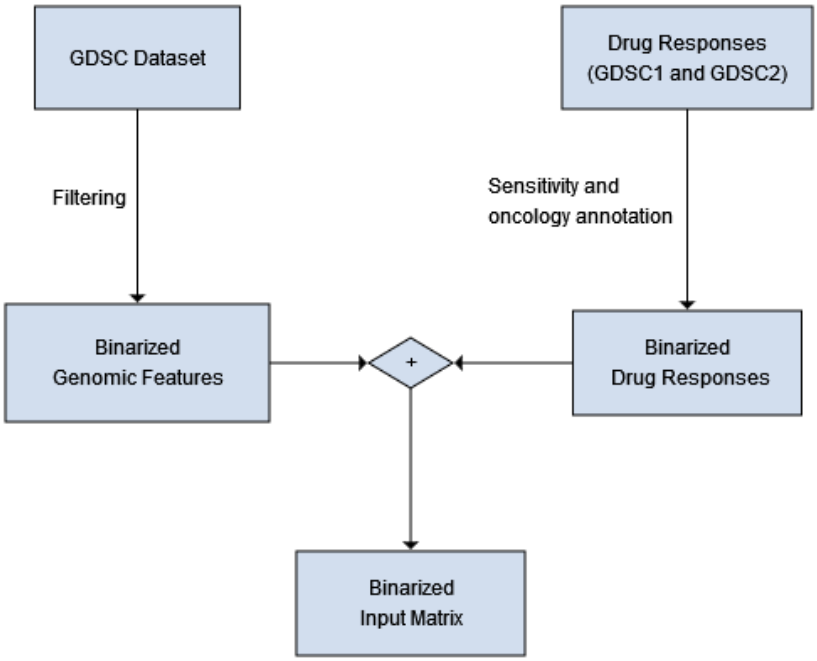

Figure 1: A simplified summary of the MERIDA data generation workflow utilizing the datasets described above as input. The output is a binarized N x P matrix consisting of N cell lines and P genetic features.

matrix as a new column **class_label**, assigning a class (sensitive, resistant) to each cell line with a 1 and 0, respectively.

KDM6A_Unknown	HNF1A_Loss-of-function	...	ARFGAP3_low_expr	class_label
0	0	...	0	0
0	1	...	0	0
0	0	...	0	1
0	0	...	0	0
0	0	...	1	1

Table 1: Excerpt from the input matrix for the drug Rapamycin. The column names describe annotated mutation features and gene expression information. The values show whether the features are present (1) or not (0) in a cell line. The class label indicates drug response (1=sensitive, 0=resistant).

MERIDA can work with or without *a priori* integration. If no prior knowledge is requested or not available, a matrix is constructed like outlined before. However, if *a priori* inclusion is desired, then all features with known sensitivity associations are merged together as one of four composite features, depending on whether they either are associated with sensitivity or associated with resistance. A second matrix containing these features is constructed. The composite features are used to enable *a priori* inclusion in the ILP implementation.

Since we are using the same data as well as data generation process as Lenhof et al. [3], we were expecting identical feature matrices to be constructed. However, despite continued

correspondence, we were unable to reproduce the exact same input data as the MERIDA authors. With the reason being unclear, our input matrices contain slightly deviating feature sets on some drugs compared to the original paper and the updated values by Lenhof. We show this slight differences in Table 2.

	Features (with a priori)		Features (without a priori)	
Drug (GDSC version)	Result	MERIDA	Result	MERIDA
Alpelisib (GDSC2)	1479	1481 (1481)	1479	1481 (1481)
Apitolisib (GDSC1)	1494	1497 (1497)	1493	1496 (1496)
AZD8186 (GDSC2)	1474	1474 (1474)	1474	1474 (1474)
CX-5461 (GDSC1)	1496	1502 (1500)	1497	1503 (1501)
Dactolisib (GDSC2)	1475	1475 (1481)	1474	1474 (1480)
MK-2206 (GDSC2)	1477	1480 (1480)	1476	1479 (1479)
Rapamycin (GDSC2)	1470	1471 (1474)	1476	1477 (1480)
Pictilisib (GDSC2)	1477	1477 (1477)	1478	1478 (1478)
Talazoparib (GDSC2)	1474	1473 (1479)	1475	1474 (1480)
Taselisib (GDSC2)	1479	1481 (1481)	1479	1481 (1481)
Temsirolimus (GDSC1)	1494	1501 (1500)	1496	1503 (1502)

Table 2: Feature size comparison of selected drugs between updated MERIDA (originally published values in parentheses) and our results.

3.2 LOBICO

LOBICO constructs its logic formulas in disjunctive normal form. Its objective function minimizes the prediction error.

$$\min \sum_{\forall c_n : y_n = 0} w_n y'_n - \sum_{\forall c_n : y_n = 1} w_n y'_n \tag{4}$$

First, LOBICO constructs K terms $t_1, ..., t_K$ by conjunction of all input features $s_1, ..., s_p$ and their negations $s'_1, ..., s'_p$ for each cell line. Then, the prediction vector y' is generated from the disjunction of these K terms for each cell line.

LOBICO has two hyperparameters, which describe the number of literals s_p and s'_p per term (M) and the number of terms t_k (K).

3.3 MERIDA

Like LOBICO, MERIDA's objective function minimizes the prediction error.

$$\min \sum_{\forall c_n : y_n = 0} w_n y'_n - \sum_{\forall c_n : y_n = 1} w_n y'_n \tag{5}$$

MERIDA assumes that cell line is only considered sensitive towards a drug if sensitivity-associated and no resistance-associated features are present. Therefore, MERIDA first collects all features a associated with sensitivity into a set s and all features b associated with resistance into a set r for each cell line:

$$s = (a_1 \vee ... \vee a_p) \tag{6}$$

$$r = (b_1 \vee ... \vee b_p) \tag{7}$$

Then, according to the prediction assumption, MERIDA obtains the final prediction vector for a cell line n:

$$y'_n = s_n \wedge \neg r_n = (a_1 \vee ... \vee a_p) \wedge \neg(b_1 \vee ... \vee b_p) \tag{8}$$

The following function with $v \in \{1, 2, 3\}$ describes the linear, quadratic and cubic weight functions respectively.

$$w_n = \frac{|Y_n - t|}{2 \sum_{\forall c_m : y_m = y_n} |Y_m - t|^v} \tag{9}$$

MERIDA only has one hyperparameter, the model size M, which is the number of features that are considered for s and r.

4 Results

To encourage flexibility and reproducibility, we implemented a Snakemake [11] workflow to run MERIDA and LOBICO experiments with configurable parameters. Snakemake is a workflow management system designed to create reproducible and scalable data analysis and is based on a Python-like language. We use Python 3 and Gurobi [12] as our optimizer of choice since it offers a Python interface and integrates well into our workflow.

Tran kindly provided us with a Snakemake template as well as a Python LOBICO implementation used in his master thesis [13]. Our Snakemake pipeline consists of rules which define input and output files based on configuration parameters. This allows for a deterministic sequence of processes to be executed in parallel.

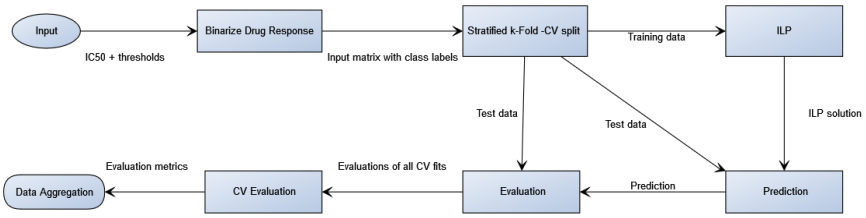

Figure 2: Overview of our Snakemake workflow. This figure depicts each major process and data flow.

We focused mainly on replicating two experiments performed by Lenhof et al. [3], a runtime analysis and statistical performance analysis. Both experiments compare LOBICO and MERIDA in terms of performance as well as the implication of utilizing different weighting functions on the outcome. All experiments were performed using a stratified 5-fold cross-validation and a wall time of 8 hours.

We ran LOBICO and MERIDA on HILBERT, the high performance computing system of the Heinrich Heine University, which is an Intel Xeon E5-2697 Ivy-Bridge processor with 24 cores and 128 GB RAM.

Since Lenhof et al. [3] greatly focused on mTOR pathway inhibitors, we decided to analyze two mTOR inhibitors (Rapamycin, Dactolisib) and two drugs (CX-5461, Niraparib) with a high number of sensitive cell lines used in the MERIDA manuscript. Additionally, we chose two drugs not part of the original MERIDA experiments (THZ-2-102-1, Gemcitabine) that show the highest standard deviation among the GDSC IC50 drug responses.

4.1 Runtime Analysis

We investigated the runtimes and memory consumption of LOBICO with ($M = 4, K = 1$), ($M = 1, K = 4$), ($M = 2, K = 2$), and MERIDA with a maximum model size of 8 ($M = 2$, $M = 4$, $M = 8$). In contrast to Lenhof et al. [3], who processed another input matrix processed differently for this experiment, we used the already generated input data without *a priori* from Section 3.1. Analogous to Lenhof et al. [3], we removed gene expression data, keeping only mutation and CNV features, and constructed new input matrices sizing from 50 up to 400 features with

Drug	GDSC version	Binarization Threshold
Rapamycin	GDSC2	-4.25527488930292
Dactolisib	GDSC2	-3.39778231725743
Niraparib	GDSC2	4.0375589217371
CX-5461	GDSC1	3.57058254269966
THZ-2-102-1	GDSC1	-4.72637365735214
Gemcitabine	GDSC2	-4.19309096257369

Table 3: A list of drugs and their calculated binanrization thresholds used in the following experiments. The latter two do not appear in MERIDA experiments.

a step-size of 50. As we applied a 5-fold cross validation, each parameter variation is executed five times. The means are shown in the graphs below, excluding experiments with a runtime exceeding eight hours.

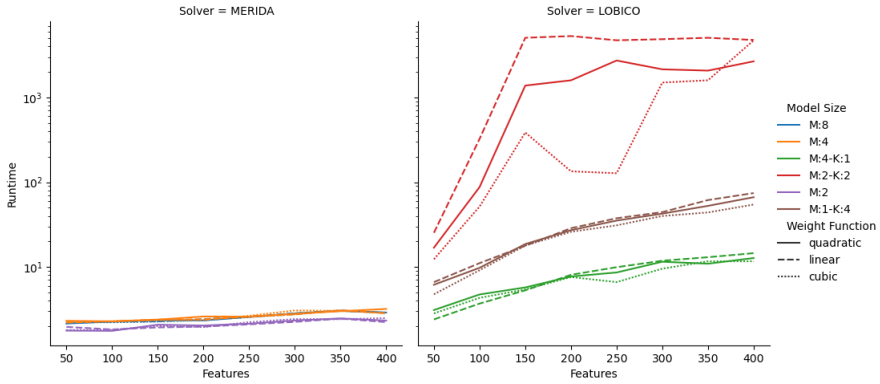

Figure 3: Average runtime in seconds of all analyzed drugs for an input matrix with 50 up to 400 features, grouped by weight function and model size.

First, we see that MERIDA has a drastically reduced runtime compared to LOBICO, which only increases minimally in proportion to the number of features. MERIDA shows an average runtime of 2.54 seconds for linear and 2.58 seconds for both quadratic and cubic weight functions. LOBICO averages around 861.27 seconds for the linear, 491.77 seconds for the quadratic and 319.03 seconds for the cubic weight function. It seems that different weight functions do not have an influence on MERIDA. LOBICO on the other hand shows improved average runtime in utilizing a cubic and quadratic weight function compared to a linear one. This is especially discernible with model parameters $K = 2, M = 2$. A higher feature size appears to affect the runtime of LOBICO more than MERIDA.

In addition to runtime, we decided to also take a look at memory usage.

We see that MERIDA needs less than 100 MB RAM in general while LOBICO with $K = 4, M = 1$ and $K = 1, M = 1$ consumes slightly more memory. There seems to be no noticeable difference between the weight functions. LOBICO's $K = 2, M = 2$ model on the other hand uses up more than 10 times more RAM than the other configurations, with the cubic and

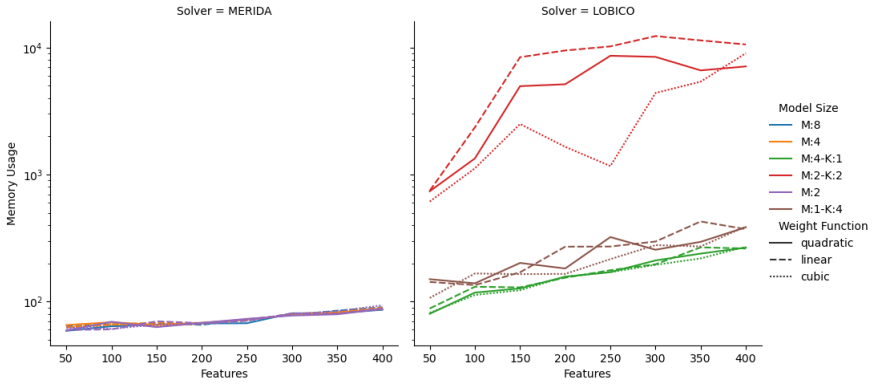

Figure 4: Average memory consumption in MB of all analyzed drugs for an input matrix with 50 up to 400 features, grouped by weight function and model size.

quadratic weight function consuming less memory than the linear function in general.

4.2 Statistical Performance

We replicate the experiment performed by Lenhof et al. [3] in using three different settings in combination with linear, quadratic and cubic weight functions. We also do not include experiments that did not finish within the specified wall time of 8 hours, similar to the original MERIDA experiments.

1. Setting 1: no a priori knowledge is included
2. Setting 2: a priori knowledge is included, and the composite features are fixed to 1
3. Setting 3: a priori knowledge is included, but the ILP determines the corresponding values

Since no *a priori* functionality is implemented in LOBICO, we will test it only with Setting 1 and 3. Unfortunately, there seems to be no available a priori knowledge on THZ-2-102-1 and Gemcitabine in our datasets. Therefore, we can only include experiments on these drugs with Setting 1.

4.2.1 Specificity and Sensitivity

We evaluate the test sets and prediction error based on sensitivity and specificity. This metrics were chosen for easier comparison to the results of Lenhof et al. [3]

LOBICO shows overall high specificity with a mean of 0.90 and sensitivity averaging 0.42. Contrarily to runtime, no weight function seems to over- or underperform significantly in comparison.

MERIDA shows lower specificity, around 0.49, than LOBICO but has a higher average sensitivity of 0.67. Again, there is no significant difference between the weight functions observable.

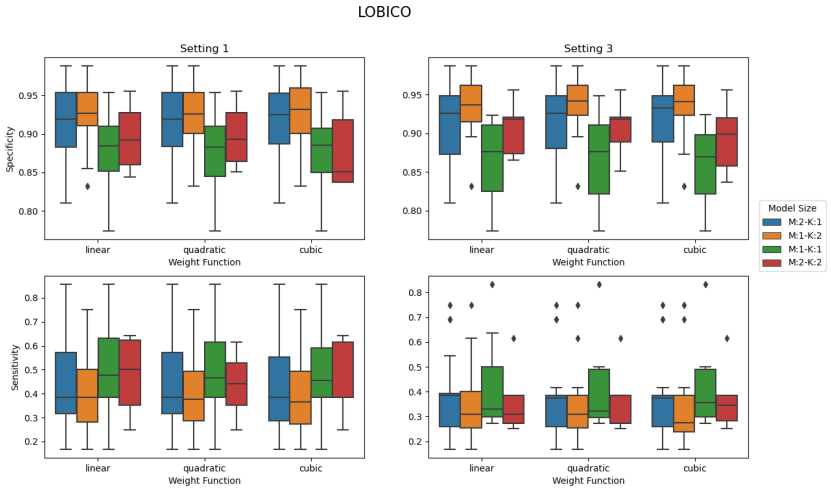

Figure 5: Specificity and sensitivity of LOBICO averaged across all analyzed drugs using a model size of 4, grouped by weight functions.

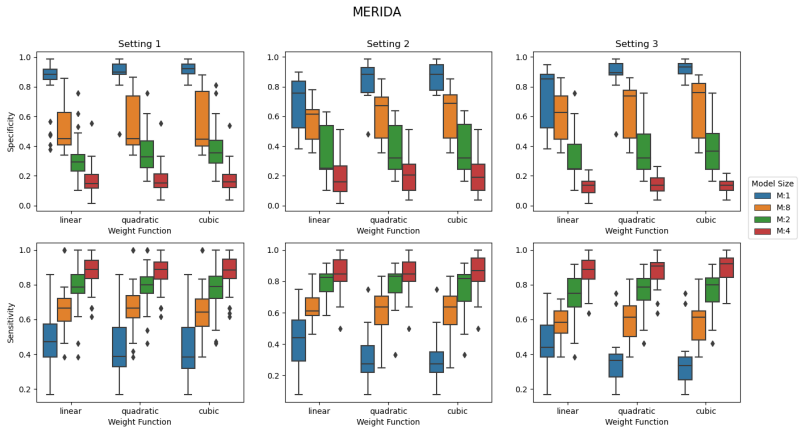

Figure 6: Specificity and sensitivity of MERIDA averaged across all analyzed drugs with a model size up to 8, grouped by weight functions.

However, smaller model sizes seem to score higher on the specificity metric while larger models perform well on the sensitivity metric. Furthermore, Setting 2 seems to have no noticeable effect on sensitivity or specificity for any weight function, regardless of model size.

Additionally, we compare the results of drugs originally used in the MERIDA experiments (see Fig. 7 and Fig. 8) to the statistical performance of our chosen drugs (see Fig. 9 and Fig. 10).

There seems to be no major difference between both drug sets, although the data for Gemcitabine and THZ-2-102-1 show a higher average sensitivity for LOBICO at 0.58 and MERIDA at 0.77 in Setting 1, respectively.

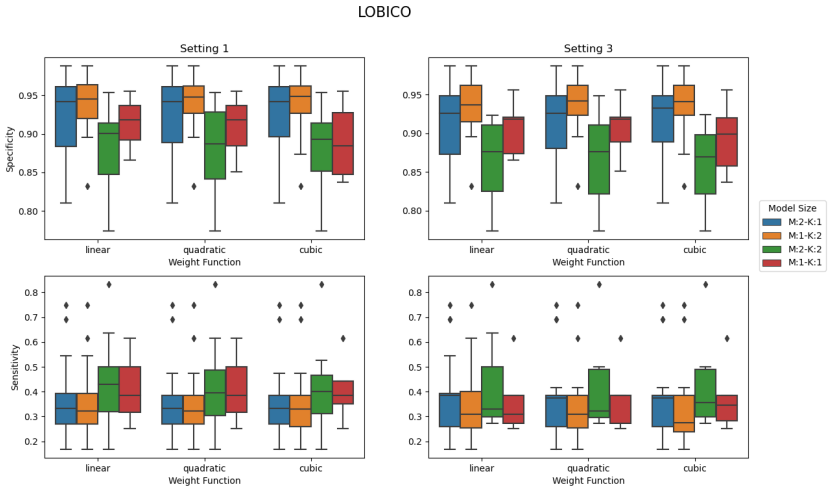

Figure 7: Specificity and sensitivity of LOBICO using a model size of 4 for Rapamycin, Dactolisib, Niraparib and CX-5461, grouped by weight functions.

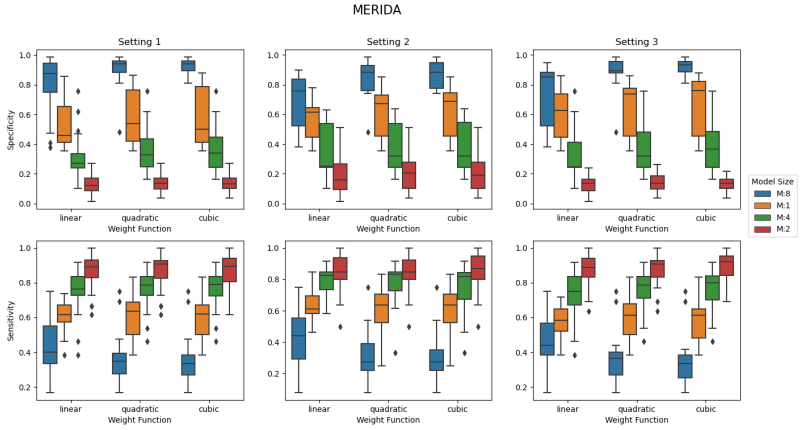

Figure 8: Specificity and sensitivity of MERIDA using a model size up to 8 for Rapamycin, Dactolisib, Niraparib and CX-5461, grouped by weight functions.

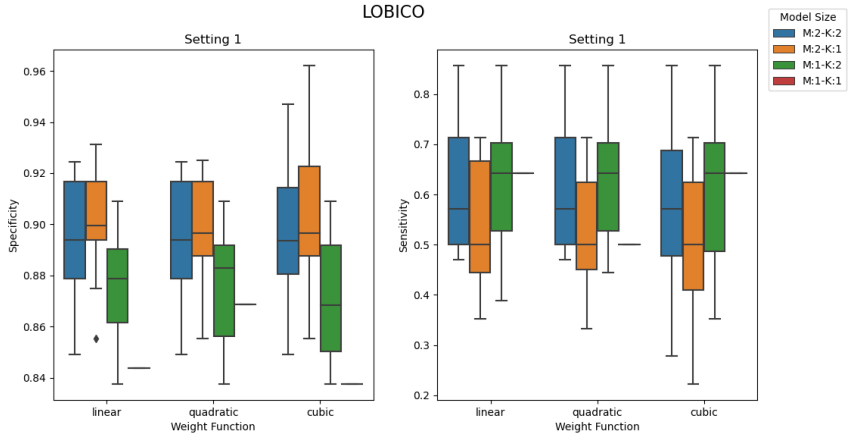

Figure 9: Specificity and sensitivity of LOBICO using a model size of 4 for Gemcitabine and THZ-2-102-1, grouped by weight functions.

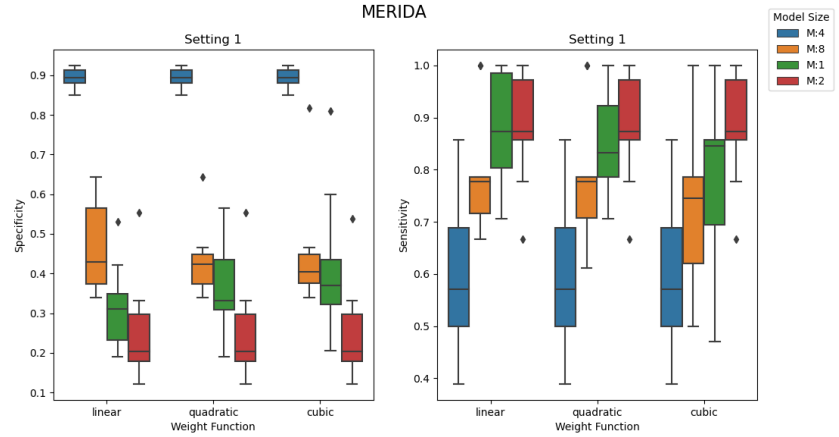

Figure 10: Specificity and sensitivity of MERIDA using a model size up to 8 for Gemcitabine and THZ-2-102-1, grouped by weight functions.

5 Discussion

We will critically analyze our methods and results in this section.

5.1 Data and Preprocessing

Unfortunately, the data generation workflow is complex and hard to understand for those uninvolved in MERIDA. At first, we had difficulties in generating input data since adjusting parameters and executing the scripts in the correct order is very time-consuming. To overcome this problem and reduce the complexity of the workflow, we have written a wrapper script that has three parameters: a drug name, GDSC version and boolean value dictating whether a priori should be included or not. The wrapper script then automatically calls every required script in order and passes the required parameters to each, utilizing an editable config file. This reduces the complexity and required user interaction time considerably and may be improved further by converting this workflow to a Snakemake pipeline.

However, MERIDA is, for now, unpractical in application. The current process relies on specific dataset versions, which Lenhof et al. [3] have described in detail, but some, like *COSMIC* [6] and *oncoKB* [7], have since been updated to a new format and are incompatible with MERIDA currently. Older versions seem not to be available for download anymore.

5.2 Methods

It would be interesting to see how LOBICO performs in Setting 2, but we were unable to implement this feature due to time constraints.

Also, *a priori* knowledge is only available for a select few drugs in our dataset, so results concerning Setting 2 and 3 might not be very accurate.

5.3 Results

We have excluded experiments exceeding the set wall time of 8 hours. This has heavily impacted results concerning LOBICO's $M = 2, K = 2$ model. Instead, we could have included the best result LOBICO has found during the allowed runtime by setting up a time limit for Gurobi. Unfortunately, we were unable to repeat affected experiments due to time constraints.

5.3.1 Runtime Analysis

As shown earlier, LOBICO seems to benefit from quadratic and cubic weight functions, especially the computationally demanding $M = 2, K = 2$ configuration. A reason might be that since cubic and quadratic weight functions decrease the weight in itself and the objective function of the ILP is a minimizing one, the objective value of cell lines is smaller the further away from the threshold they are, and a solution might be found faster.

When comparing a model size of 4 and using a linear weight function, MERIDA with $M = 4$ is on average 3.0 times faster than $M = 4, K = 1$, 21.36 times faster than LOBICO's $M = 1, K =$

4 and 1241.02 times faster than $M = 2, K = 2$ configuration. Using a quadratic or cubic weight function on LOBICO's $M = 2, K = 2$ shows an improved runtime by a factor of 2.64 and 6.96, respectively.

Our results show a high standard deviation for LOBICO runtimes on a smaller dataset, although they are in line with Lenhof et al. [3]

5.3.2 Statistical Performance

In our results, MERIDA has a lower average specificity than LOBICO, but performs well regarding sensitivity. Interestingly, larger MERIDA model seem to increase sensitivity while smaller models have better specificity, although it is not apparent in our data whether *a priori* knowledge improves sensitivity. This interpretation might be obstructed by the high standard deviation our results show and by the small sample size of drugs with available *a priori* knowledge.

Generally, our results are similar to Lenhof et al. We agree that a higher sensitivity might be a good choice for a model working with unknown data. Since MERIDA is very fast, further analysis on performance of larger models that might be better suited to represent the biological complexity would be feasible.

6 Conclusion

In this thesis, we analyzed MERIDA [3] regarding credibility and reproducibility. MERIDA claims to be faster than LOBICO, comparing a model with a feature size of 4 and the same weighting function. Additionally, Lenhof et al. [3] results show that LOBICOs runtime can be improved by utilizing a different weighting function instead of a linear one. Compared to LOBICO, Lenhof et al. show that MERIDA has a lower average specificity but a higher overall sensitivity. A priori knowledge (Setting 2) should increase the sensitivity slightly.

Although we had difficulties reproducing input data used by the ILP at first, our results show that MERIDA ($M = 4$) is indeed faster than LOBICOs 4-feature sized models (($K = 1, M = 4$), ($K = 4, M = 1$), ($K = 2, M = 2$)) by up to 1241.02 times on average. LOBICOs runtime with parameters $K = 2, M = 2$ show an improvement by a factor of up to 6.96 using a cubic weight function, according to our experiments. Our data confirms that MERIDA demonstrates a higher average sensitivity than LOBICO while showing lower specificity. Additionally, smaller MERIDA models seem to score higher on specificity and lower on sensitivity while larger models have lower specificity but high overall sensitivity. Our results do not seem to indicate that different weight functions affect these metrics.

In conclusion, our results based on the reproduced data matrices seem promising and also support the results Lenhof et al. [3] show in their manuscript.

References

[1] Delora Baptista, Pedro G Ferreira, and Miguel Rocha. “Deep learning for drug response prediction in cancer”. In: *Briefings in Bioinformatics* 22.1 (Jan. 2020), pp. 360–379. ISSN: 1477-4054. DOI: 10.1093/bib/bbz171. eprint: https://academic.oup.com/bib/article-pdf/22/1/360/35934938/bbz171.pdf. URL: https://doi.org/10.1093/bib/bbz171.

[2] Theo A Knijnenburg et al. “Logic models to predict continuous outputs based on binary inputs with an application to personalized cancer therapy”. In: *Scientific reports* 6.1 (2016), pp. 1–14. DOI: 10.1038/srep36812. URL: https://doi.org/10.1038/srep36812.

[3] Kerstin Lenhof et al. “MERIDA: a novel Boolean logic-based integer linear program for personalized cancer therapy”. In: *Bioinformatics* 37.21 (Aug. 2021), pp. 3881–3888. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btab546. eprint: https://academic.oup.com/bioinformatics/article-pdf/37/21/3881/41091860/btab546.pdf. URL: https://doi.org/10.1093/bioinformatics/btab546.

[4] Francesco Iorio et al. “A Landscape of Pharmacogenomic Interactions in Cancer”. In: *Cell* 166.3 (2016), pp. 740–754. ISSN: 0092-8674. DOI: https://doi.org/10.1016/j.cell.2016.06.017. URL: https://www.sciencedirect.com/science/article/pii/S0092867416307462.

[5] Malachi Griffith et al. “CIViC is a community knowledgebase for expert crowdsourcing the clinical interpretation of variants in cancer”. In: *Nature genetics* 49.2 (2017), pp. 170–174.

[6] John G Tate et al. “COSMIC: the Catalogue Of Somatic Mutations In Cancer”. In: *Nucleic Acids Research* 47.D1 (Oct. 2018), pp. D941–D947. ISSN: 0305-1048. DOI: 10.1093/nar/gky1015. eprint: https://academic.oup.com/nar/article-pdf/47/D1/D941/27441712/gky1015.pdf. URL: https://doi.org/10.1093/nar/gky1015.

[7] Debyani Chakravarty et al. “OncoKB: A Precision Oncology Knowledge Base”. In: *JCO Precision Oncology* 1 (2017), pp. 1–16. DOI: 10.1200/PO.17.00011. eprint: https://doi.org/10.1200/PO.17.00011. URL: https://doi.org/10.1200/PO.17.00011.

[8] D Tamborero et al. “Cancer Genome Interpreter annotates the biological and clinical relevance of tumor alterations. Genome Med 10 (1): 25”. In: (2018).

[9] Abel Gonzalez-Perez et al. “IntOGen-mutations identifies cancer drivers across tumor types”. In: *Nature methods* 10.11 (2013), pp. 1081–1082.

[10] Francisco Sanchez-Vega et al. “Oncogenic Signaling Pathways in The Cancer Genome Atlas”. In: *Cell* 173.2 (2018), 321–337.e10. ISSN: 0092-8674. DOI: https://doi.org/10.1016/j.cell.2018.03.035. URL: https://www.sciencedirect.com/science/article/pii/S0092867418303593.

[11] Mölder F, Jablonski KP, Letcher B et al. *Sustainable data analysis with Snakemake.* 2021. DOI: https://doi.org/10.12688/f1000research.29032.1.

[12] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. 2022. URL: https://www.gurobi.com.

[13] Nguyen Khoa Tran. "Logic models to classify cancer types based on tumor DNA samples". MA thesis. Heinrich Heine University, 2021. URL: https://www.cs.hhu.de/fileadmin/redaktion/Fakultaeten/Mathematisch-Naturwissenschaftliche_Fakultaet/Informatik/Algorithmische_Bioinformatik/Bachelor-_Masterarbeiten/2436588_ms_ifo_AbschlArbeit_klau_marscht_ngtra102_20210105_1914.pdf.