Chapter 1

Group Sampling

1.1 Introduction

Group sampling is an experimental design introduced to handle large parameter sets in a sensitivity analysis [**?**]. This design allows an analyst to identify influential parameters and determine their influence. It even allows obtaining sensitivity analysis information from so-called supersaturated designs. These are designs, where the number of measurements is smaller than the number of parameters.

The assumption is made, that the influence of each parameter is negligible. This assumption is rejected if there is data providing strong evidence of this influence. With this approach, the problem can be viewed as one of statistical testing [**?**]. This makes it possible to test for influential parameters and later reuse this information to analyze the nature of these influences. At the core of group sampling is the idea, that information about parameters can be extracted if multiple parameters are put in a group and tested together.

How the amount of testing can be reduced by clever designs, can be explained by recent efforts to reduce the number of tests needed to test as many people as possible for the SARS-CoV-2-Virus. **?**] describes several approaches for pooled testing.

Here, a slightly altered version will be explained to more easily show, how the number of tests can be reduced by pooled testing. We assume we want to test 9 individuals for SARS-CoV-2 and want to reduce the number of tests needed. We also assume that the chance of more than one person being infected is negligibly small. By grouping our 9 individuals into groups and only using one test for the whole group, we can reduce the needed tests without significantly impacting the chance of detection of an infected individual.

This can be done by grouping these 9 individuals in a specific way. First, we create a matrix with the same amount of elements as the individuals to test and

the dimensions $m \times n$. In this case, a 3x3-matrix. Now we assign individuals to a corresponding place in the matrix. This is done by assigning the element x_{ij} to the individual $I_{(mi+j)}$. The groups for testing are then created by taking each column and each row of the matrix as a group, resulting in six groups total. For ease of reference, the groups created from the rows are g_i where i is the index of the row and the groups created from the columns are h_j where j is the index of the column. This form of grouping gives us a way to re-identify an infected person if there is one. Let's assume I_5 is infected with SARS-CoV-2 in this example. Subsequently, the groups containing this individual would test as positive. In our example, if I_5 is infected, the groupings g_2 and h_5 would test as positive. To re-identify the individual, the overlap between those two groupings needs to be identified. This can be done by looking at the indices of the groupings $f(g_i, h_i) = m * i + j$.

With this, the amount of needed tests to test 9 individuals is reduced from 9 to 6 tests. This design, in theory, can be scaled up but may suffer in reliability when increased in size. In the case of the SARS-CoV-2 example, the tests used on the group might not be able to identify if a group is infected if the amount of individuals in this group is higher than a certain threshold.

The same idea of extracting as much information as possible out of groupings is used in group sampling. Here, the parameters are grouped randomly in groups of the same size. For each group, a test is done, and the influence values are stored. By repeating this grouping multiple times with different groups, information about the influence of a single parameter can be extracted.

1.2 Group sampling for sensitivity analysis

?] describes how group sampling can be applied to perform sensitivity analysis on a given model. Suppose we have a model with 1000 parameters X_i with $i \in [1 - 1000]$ where only a few parameters are influential. We group the parameters into groups of equal size. With 1000 parameters a sensible amount of groups could be M=10, giving us 10 groups containing 100 parameters. Group G_1 would be $G_1 = \{X_1, X_2, X_3, ..., X_{100}\}$, $G_2 = \{X_{101}, X_{102}, X_{103}, ..., X_{200}\}$ and Group $G_{10} = \{X_{901}, X_{902}, X_{903}, ..., X_{1000}\}$. This kind of grouping would be repeated N times, with randomly assigned parameters for each group, resulting in N * M groups $g_{n,m}$, where for each grouping $g_{n,M}$ all sets of parameters are distinct. Each group is now treated as a parameter of its own, assigning all parameters in a group the same value, giving us an abstraction of the model with only 10 parameters. Now, on these 10 groups, we can perform a set of simulations to determine the influence of the group as if it were a single parameter.

Table 1.1: Parameter groupings and their influences

M	G_1	G_2	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
1	**8.1**	1.8	**8.1**	1.8	1.8	**8.1**	1.8	**8.1**	1.8	1.8	**8.1**	**8.1**
2	**9.3**	5.1	**9.3**	5.1	5.1	**9.3**	5.1	**9.3**	**9.3**	5.1	**9.3**	5.1
3	**10**	6.5	6.5	6.5	6.5	**10**	6.5	**10**	**10**	6.5	**10**	**10**
4	**9.8**	5.4	5.4	**9.8**	5.4	**9.8**	**9.8**	**9.8**	**9.8**	5.4	5.4	5.4
5	**4.7**	0.9	**4.7**	0.9	**4.7**	**4.7**	**4.7**	0.9	0.9	**4.7**	0.9	0.9
			6.8	4.8	4.7	8.4	5.6	7.6	6.4	4.7	6.7	5.9

In Table 1.1 an example with 10 parameters, two groups M=2 and 5 rounds of random grouping can be found. All parameters with bold font are in the first group of the grouping M. For each group an influence value is determined and each parameter in the group is assigned this influence value for this grouping. If we look at the second grouping, the parameters X_1, X_4, X_6, X_7, X_9 are in this first group G_1 and have the influence value of 9.3. The parameters X_2, X_3, X_5, X_8, X_{10} are in the second group G_2 and have the influence value of 5.1. In this example, the parameter X_4 is an influential parameter. We can already see, with the second grouping alone, the influential parameter is most likely contained in group G_1 for the second grouping. If we take the average value of each parameter for each grouping, we can estimate the influence of each parameter. In this example, we can see the parameter X_4 has the highest influence and is most likely our influential parameter.

From this example, we can also see how non-influential parameters can look influential if they share the group with the influential parameter too often. If we look at X_6, the parameter looks influential, even though if the actual influential parameter is not in the same group, the group influence is small. This makes it harder to determine the actual influential parameter. **?**] give us the probability of two parameters, X_i and X_j sharing a group t times with:

$$P_{ij}(t, N, S) = \binom{N}{t} \left(\frac{1}{S}\right)^t \left(\frac{S-1}{S}\right)^{N-t} \tag{1.1}$$

For our example with N = 5 groupings and a group size of $S = \frac{|X|}{M} = 5$, this gives us a probability of X_4 and X_6 sharing the same group t=3 times with $P_{4,6}(3, 5, 5) = 0.051$ and a total probability of any parameter sharing the group with the influential parameter X_4 t times with $1 - (1 - P_{ij}(3, 5, 5))^9 = 0.377$.

1.3 Group sampling with configuration options

In order to implement group sampling on configuration options, the method described by **?** needs to be adapted. For simplicity, we only look at feature models with binary features. When grouping features, each group of features needs to be a valid configuration if all features in a group are enabled. Otherwise, measuring the influence of the group on its own is not possible. This forces us to adhere to the constraints on the feature model while creating groups. We can identify three different types of features, which affect the way we can assign features to groups.

Mutually exclusive features

In this work, we call a group of features, where we can not enable more than one at the same time, a group of mutually exclusive features and consequently a feature contained in one of those groups, a mutually exclusive feature. For example, an alternative group is always a group of mutually exclusive features, since only one feature can be enabled without violating the constraints. Mutually exclusive features are problematic when we group features. We can't have two features assigned to the same group if they are together in a mutually exclusive group. This would cause an invalid configuration once the features of the group are enabled.

Independent features

We call features, which do not have any constraints on them and are optional, independent features. If a set of features already are a valid configuration, these features can be added or removed completely independent of the already selected features. The resulting set of features would still be a valid configuration. These features allow for the easy creation of groups among them since they can be combined in any way possible without violating any constraints.

Implying feature

Implying features, or as **?** categorizes them, requires features [**?**], are features with a constraint, which forces us to select the implied feature if the implying feature is enabled. If an implying feature is assigned to a group, we need to assign the implied feature to the same group, otherwise, we would get an invalid configuration if the group with the implying feature is selected and the one with the implied feature is not.

In **?**] group sampling, each group is a distinct set of parameters. This limits the number of groups possible on a set of features with constraints. While mandatory features would make it impossible to create any groups, we simply could ignore them, since they do not impact the performance of a given system. The most limiting factor would be a mandatory group of mutually exclusive features. The number of possible groups with a distinct set of features would equal the smallest mandatory group of mutually exclusive features. If a feature model contains multiple sets of mutually exclusive groups of different sizes, it is impossible to assign all mutually exclusive features to a group while still having a distinct set of features for each group.

To create a performance influence model we need to measure the influence of each group individually. By doing so, we can estimate the influence of each group of features. With repeated testing of different groupings, we can estimate the influence of each feature by averaging the influence of the feature across all groupings. While this does not give us a perfect estimate of the influence of each feature, it lets us test, which of the features is most likely influential and to what degree.

1.4 Creation of configuration groups

In this section, we tackle the problem of adapting the method described by **?**] to configuration options. We especially try to handle the problem of constraints among the features during group creation. We devise two strategies to create groups on features and handle the complexity of constraints during group creation. In the previous section, we identified several problems the constraints cause when creating groups of features. Both methods described in the following sections aim to mitigate these obstacles in order to still create groups in the presence of constraints.

1.4.1 Maximizing Hamming distance

This method aims to generate groupings of features without any previous analysis of the feature model. The main idea is to create groups of features with minimal overlap between them. In the best-case scenario of a feature model with only independent features, this would create groups without any overlap between them and thus completely distinct sets of features. To measure the overlap between two groups of features we use the Hamming distance between

the two groups.

$$D_H = \sum_{i=1}^{n} f(A_i, B_i)$$
$$f(A, B) = \begin{cases} 0 & \text{if } A = B \\ 1 & \text{else} \end{cases} \tag{1.2}$$

This means, the Hamming distance between two groups is equal to the amount of features which are not in both groups.

Since we do not have a hard constraint of no overlap between groups, mandatory features are allowed to be in all groups. This allows us to ignore them during group creation due to the fact, that their influence on performance does not contribute to performance variation. The complexity of mutually exclusive groups and features with implications is implicitly solved by the SAT-Solver, since it is responsible to create valid configurations.

We also need to define how many features should be contained in a group. Otherwise, we leave it open to the SAT-Solver to determine the number of features in a group, which would result in very uneven group sizes due to the nature of SAT-Solvers. Since we do not have any previous knowledge of the number of independent features, mutually exclusive groups or features with implications, the most straightforward answer to the number of features which should be contained in a group would be:

$$Features\ in\ group = \left\lfloor \frac{No.\ Features}{Group\ size} \right\rfloor \tag{1.3}$$

1.4.2 Grouping independent features

While grouping features using the Hamming distance between the groups simplifies the process of creating groups by pushing the complexity to the SAT-Solver. This method provides a more hands-on approach. We analyze the feature model and determine the independent features and the groups of mutually exclusive features. With the knowledge we have, we can more easily create valid groups of features.

Groups with mutually exclusive features

To create groups among mutually exclusive features, we want to pick one feature out of each group of mutually exclusive features, if possible. With this, we get the maximal amount of features we can group together across all mutually exclusive groups. If we assume, we have a feature model with two alternative groups, the maximal amount of features, we can group together is

two, since only one feature out of each alternative group can be selected. To look for mutually exclusive features in a feature model, we can use the SAT-Solver. By enabling each combination of any two features and checking, if there is a valid configuration, we can find out which two features are mutually exclusive. The pairs of mutually exclusive features help us to determine the mutually exclusive groups. In a mutually exclusive group, only one feature of the group can be enabled. This means all features in the group are mutually exclusive with all other features in the group. If we translate the mutually exclusive relationship between features into an undirected graph, where the nodes represent the features and the edges the mutually exclusive relationship, we can reformulate the problem of finding the groups as the clique problem [**?**]. The cliques in the graph represent a mutually exclusive group because all nodes in the clique are fully connected. Figure 1.1 depicts such a graph.

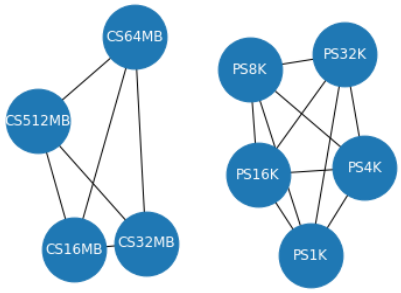

Figure 1.1: Graph of mutually exclusive features in the BerkeleyDB dataset.

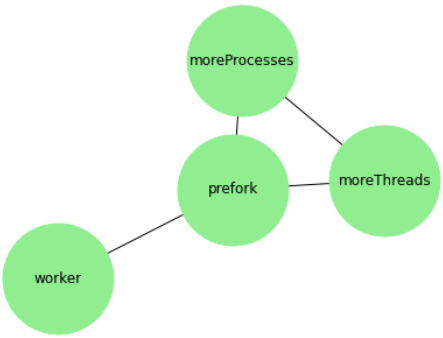

Figure 1.2: Partial graph of mutually exclusive features in the Apache dataset.

When we select features out of mutually exclusive groups we have to be aware, that mutually exclusive groups can overlap. This presents a problem

when we select an overlapping feature. To illustrate the problem, a subgraph from the mutually exclusive relationships in the Apache dataset is presented in Figure 1.2. The mutually exclusive groups, the cliques, in this example would be *worker, prefork* and *prefork, moreProcesses, moreThreads*. Since *prefork* is contained in both groups, if we pick this feature, we can not pick any feature of the other mutually exclusive group. To circumvent this problem, instead of choosing one feature out of each mutually exclusive group, we choose one feature out of each component[1]. This leaves us with smaller groups, since fewer features can be selected for each group, but a more equal distribution of selected features in connected mutually exclusive groups.

Groups with independent features

The independent features are all features, which can be disabled and are not in any of the mutually exclusive groups. Not all features in a feature model which are modelled as optional are really optional. **?**] describe these features, features which are included in all configurations, despite being modelled as optional, as false optional features. So to determine the independent features we need to determine features, which really can be enabled optionally. This can be done by iterating overall features and checking if a valid configuration exists where the feature is disabled [**?**]. The set of features which can be disabled without the features contained in a group of mutually exclusive features is the set of independent features. Since the independent features are not bound by any constraints, we can randomly assign a feature to a group without being concerned about violating any constraints.

We can now combine both approaches for grouping mutually exclusive features and independent features to create groups over the whole feature model. We are still limited by the fact, that we can not create more distinct groups than the amount of the smallest mutually exclusive component. This would mean, that we can't cover all features in one round of grouping. We can circumvent this problem by simply allowing for mutually exclusive features to be in multiple groups. This lets us create more groups, but makes it harder to determine the influence of a feature that is assigned to multiple groups.

To actually generate a group, we need to define how many features are in a group. The amount of features from the mutually exclusive features is simply

[1] A component of an undirected graph is connected subgraph has no edges to any other graph

the amount of components $|C|$ in the mutually exclusive graph. The amount of features in a group can be described with the following formula:

$$N = \left\lfloor \frac{No.\ independent\ features}{Group\ size} \right\rfloor \tag{1.4}$$

$$Features\ in\ group = N + |C| + mandatory\ features \tag{1.5}$$

We can see samples created from both variations in Figure 1.3. Each row represents a configuration and each column represents a feature. A feature is selected, if the block is black, otherwise it is not selected. A round of groupings is separated by a horizontal line. Looking at the figure, we can already identify a problem both varients have. The differences between the rounds of groupings are minimal, caused by the use of an SAT-Solver. We tackle this problem in subsection 1.6.2

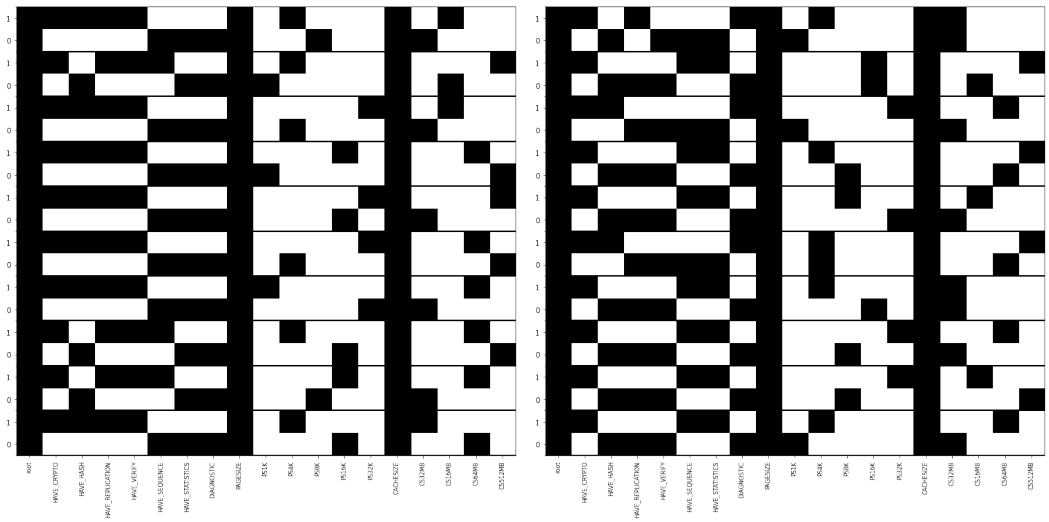

Figure 1.3: Samples generated on the BerkeleyDB Dataset by both variations to generate groups. **Left:** Group Sampling - Hemming Distance - Group Size 2 - Groupings 10 **Right:** Group Sampling - Independent features - Group Size 2 - Groupings 10

1.5 Influence model

With the groups created in the previous section, we can create a model, which predicts the performance of a given system. For this, we measure the non-functional property we want to predict for each group of features. Table 1.2 shows an example of such measurements. We only enable the features of one group at a time. The taken measurement then serves as the influence value of the group itself. In Table 1.2 each round of groupings is separated by a horizontal line. If we look at the first round of groupings, the influence value of G_1 is 3 and the features F_1 and F_2 are part of the group.

If in another measurement, the same configuration is chosen, we expect to see the influence value of the group as the measurement. Since measuring all possible groups (configurations) is not feasible, we want to determine the influence of single features to predict unseen configurations. By definition, only a few features are influential, this lets us assume that the influence measured in a grouping stems from only one or just a few features. In our example, this lets us assume that F_1 and F_2 have the influence of 3. With multiple different groupings, we can correct or confirm this assumption. The average influence value the feature has is our best approximation with the data available. If an influential feature is in a group, the group influence is most likely significantly higher. With enough groupings of different features, we can determine an influential parameter due to the fact, that the average influence of this feature is higher than that of the rest. In our example, we can see that the groups with F_6 assigned to them have, on average, a higher influence than the groups not containing F_6. With the average influence of the features, we can create a model of the system. We can use the formula for multiple linear regression as described by **?**].

nicht wirklich zufireden damit ...

$$y = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n + \varepsilon \tag{1.6}$$

We adjust the formula so that our parameters β_n represents the influence of a feature. The feature influences we determined in Table 1.2 represent the measured non-functional property of the system and include its baseline performance. This means, the performance of the system if all features would be disabled. We need to compensate for this since it would otherwise make the prediction of the model unusable. We can do this by determining the baseline performance and subtracting it from the influence values. An approximation of the baseline performance would be the average of all measurements taken. The model constructed would be described by following forumlas:

$$f_0 = \frac{1}{n} \sum_{n=1}^{|I|} I_n \tag{1.7}$$

$$f_n = I_n - f_0 \tag{1.8}$$

$$y = f_0 + f_1 x_1 + f_2 x_2 + ... + f_n x_n + \varepsilon \tag{1.9}$$

Where I_n is the average of all measurements taken which included F_n and x_n is either one, if the feature is selected or 0, if the feature is not selected. We can use this formula to predict the behaviour of the system on unseen configurations, but the accuracy of the prediction is most likely not very good. We can see why if we look at F_3 and F_5 and their respective average measurements I_3 and I_4. They both have high average values, due to the fact, that they share a group with our influential feature F_6. The effect on their influence values due to sharing a group with an influential feature would average down if the number of groupings would be increased, but it is still a problem. We try to compensate for this problem with a stepwise analysis of the influence values in subsection 1.6.1.

Table 1.2: Feature groupings and their influences

G_1	G_2	G_3	R	F_1	F_2	F_3	F_4	F_5	F_6	I_1	I_2	I_3	I_4	I_5	I_6
1	0	0	3	1	1	0	0	0	0	3	3				
0	1	0	6	0	0	1	1	0	0			6	6		
0	0	1	28	0	0	0	0	1	1					28	28
1	0	0	2	1	0	0	1	0	0	2			2		
0	1	0	7	0	1	0	0	1	0		7			7	
0	0	1	25	0	0	1	0	0	1			25			25
										2,5	5	15,5	4	17,5	**26,5**

1.5.1 Multicollinearity

In regression analysis, having correlation among predictors is undesirable [**?**]. If two or more predictors in a multiple regression model have a linear relation, it is called multicollinearity. Multicollinearity increases the standard errors and makes the coefficient unreliable, decreasing their precision [**?**]. We can use the variance inflation factor (VIF) to find multicollinearity in our data. **?**] provides us with the formula:

$$VIF_i = \frac{1}{1 - R_i^2} \; for \; i = 1, 2, ..., k \tag{1.10}$$

Where R_i^2 is the coefficient of multiple determination of independent variable x_i on the remaining variables [**?**]. We can calculate the VIF by performing a regression on all independent variables except one, for each independent variable. Each created model gives us an R^2 value, which we can use in Equation 1.5.1 to determine the VIF for an independent variable. A VIF greater than 5 indicates a high correlation [**?**].

1.5.2 Feature interactions

With group sampling, we collect information about features grouped together. This includes the interactions between features. To make use of this information, we can add the interactions of features to our calculation of the influence of individual features. A way to do this is to treat interactions similar to features during the determination of feature influences. By assigning the group value to the interaction, if and only if all features involved in the interaction are in the group, we can estimate the influence of the interaction the same way as with features. In Table 1.3 we added each interaction $I_{i,j}$ between two features F_i and F_j to the table.

Table 1.3: Feature groupings and their influences

G_1	G_2	R	F_1	F_2	F_3	I_1	I_2	I_3	$I_{1,2}$	$I_{2,3}$	$I_{1,3}$
1	0	3	1	1	0	3	3		3		
0	1	6	0	0	1			6			
1	0	19	1	0	1	19		19			19
0	1	7	0	1	0		7				
						11	5	12,5	3		**19**

$I_{1,2}$, $I_{2,3}$ and $I_{1,3}$ are the interactions between the features and get assigned the group value if both features of the interaction are selected in the group. With a smaller group size, we capture more interactions during our measurements since more features are selected in one group. We can see in our example, that the interaction between F_1 and F_3 results in a higher measurement as if the features are grouped with another feature or are in a group alone. While we were able to capture some interactions, we can see that we did not capture the interaction between F_2 and F_3. To estimate the influence of all interactions

we need a much larger sample size than to estimate the influence of features on their own.

1.6 Optimizations

1.6.1 Stepwise Influence

In section 1.5 we created a model of a system by taking the average value a feature has across groupings. In our example in Table 1.2 we were already able to see a problem **?**] described with group sampling. Features that share a group with an influential feature look more influential than they truly are. **?** describe a way to determine the influence more accurately by determining the influence of a single parameter at a time.

The idea is to remove the influence an influential feature has before determining the influence of the next influential feature. We can do this in two steps, for each feature we want to determine the influence for. First, we need to find the most influential feature we have in our data. Then we have to estimate its influence and remove it from our data.

If we look at Table 1.2 we can use the average measurements of a feature to identify the most influential. But, we have to be aware, that the average of the measurements still includes the baseline performance. By simply taking the highest measurement, we could end up picking the wrong feature, if, for example, the influential feature has a negative effect on the measurement. We want to pick the outlier value of the average feature measurements. One way to do this is to pick the feature where the average measurement is the furthest away from the average of all measurements. In Table 1.4 we can see, the average of all average measurements would be approximately 11.8, giving us F_6 as our outlier.

With the feature identified, we can remove it from our data. We estimate the influence of the feature the same way as previously, by taking the average of all its measurements. By subtracting the value from all measurements where the feature was part of the group, we can get a better estimate of the influence the rest of the features have. Then we remove the influential feature from our data and proceed with identifying the next influential feature. An example of this procedure can be found in Table 1.4. We identified F_6 as our most influential feature and removed it and its influences in other groups from the data. F_6 shared a group with F_5 and F_3 and after removing the estimated influence of F_6, both do not look as influential as in Table 1.2.

In each round, we identify the influence of one feature. With the influences of the features determined that way, we can build our model the same way as

previously, but instead of using I_n in Equation 1.5 and Equation 1.5 we use the values determined by our stepwise analysis.

Table 1.4: Stepwise influence calculation

G_1	G_2	G_3	R	F_1	F_2	F_3	F_4	F_5	F_6	F_1	F_2	F_3	F_4	F_5	F_6
1	0	0	3	1	1	0	0	0	0	3	3				
0	1	0	6	0	0	1	1	0	0			6	6		
0	0	1	28	0	0	0	0	1	1					1,5	
1	0	0	2	1	0	0	1	0	0	2			2		
0	1	0	7	0	1	0	0	1	0		7			7	
0	0	1	25	0	0	1	0	0	1			-1,5			
										2,5	5	2,25	4	**4,25**	**26,5**

1.6.2 Feature coverage

In this work, we use an SAT-solver to create valid configurations. We can see the effects of it in Figure 1.3. Off-the-shelf SAT-solvers tend to find locally clustered solutions [**?**]. The repeating patterns in our samples are a result of it. The SAT-solver finds a solution to our constraints by changing as few variables as possible. Since at the beginning of each grouping, the constraints the solver has are similar, the SAT-solver gives a similar solution. This prevents us from making meaningful groupings since features regularly share the same group.

We adapt the distance-based sampling strategy introduced by **?**] to help in creating more diverse groups. We use the Hamming distance as shown in Equation 1.2 as our distance metric and the uniform distribution as our probability distribution. Instead of measuring the distance from the origin, we measure the distance from the first group of the previously created grouping.

In detail, at the start of each grouping, we randomly pick a distance from the first group created in the previous grouping. We then set the distance as a constraint for our SAT-solver to avoid getting a similar first group. The following created groups are always dependent on the first group since they need to be distinct from each other. This way the groups created during each grouping do not follow a pattern of minimal change.

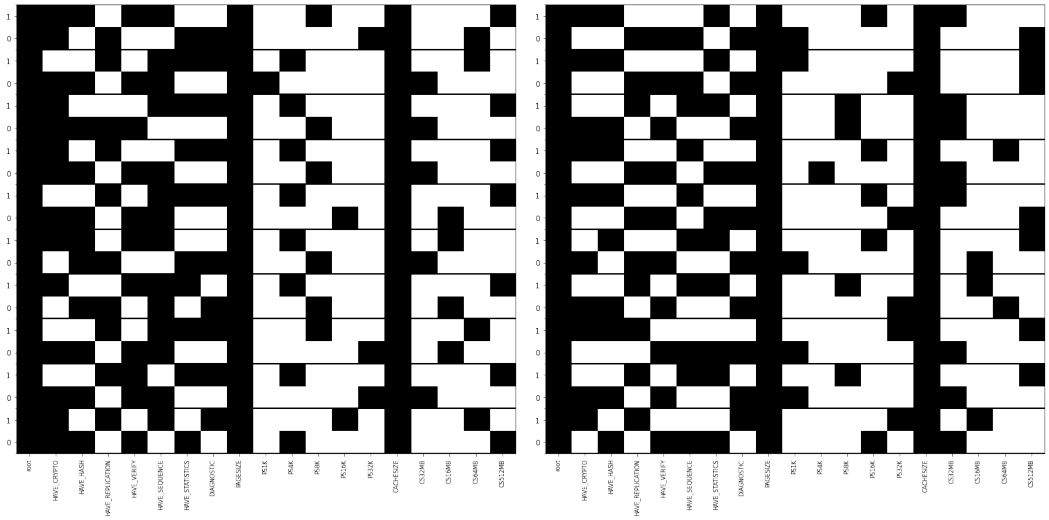

Figure 1.4: Samples generated on the BerkeleyDB Dataset by both variations with distance based group optimization. **Left:** Group Sampling - Hemming Distance - Group Size 2 - Groupings 10 **Right:** Group Sampling - Independent features - Group Size 2 - Groupings 10