Chapter 1

Group Sampling

1.1 Introduction

Group sampling is an experimental design introduced to handle large parameter sets in a sensitivity analyzes [**?**]. 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 a 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 reusing 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 as together.

How the amount of testing can be reduced by clever designs, can be explained by recent efforts to reduce the amount 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 amount 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 amount of tests needed. We also assume, that the chance of more than one person being infected is negligible small. By grouping our 9 individuals into groups and only use 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-matix. 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-identifiy 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-identifiy 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, a group size of two and 5 rounds of random grouping can be found. All parameters with a 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. **?**] gives us the probability of two parameters, X_i and X_j sharing a group t times with:

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

For our example with N = 5 grouping 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 t times with $1 - (1 - P_{ij}(3, 5, 5))^9 = 0.377$.

1.3 Group sampling with configuration options

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 influence 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 optionally, independent features. If a set 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 easy creation of groups among them, since they can be combined in any way possible without violating any constrains.

Implying feature

Implying 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 in invalid configuration if the group with the implying feature is selected and the one with the implied feature not.

In **?**] group sampling, each group is a distinct set of parameters. This limits the amount 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 amount of possible groups with a distinct set of feature would equal to 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 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 feature 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 handling 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 provide a solution to still create groups of features 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 feature with a 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 don't have a hard constraint of no overlap between groups, mandatory features are allowed to be in all groups. This allows us to simply ignore them during group creation. The complexity of mutually exclusive groups and features with implications is implicitly 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 amount 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 over the amount of independent features, mutually exclusive groups or features with implications, the most straightforward answer to the amount of feature 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 aims 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. 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 check, if it is a valid configuration, we can find out which two features are mutually exclusive. The pairs of mutually exclusive features helps 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.

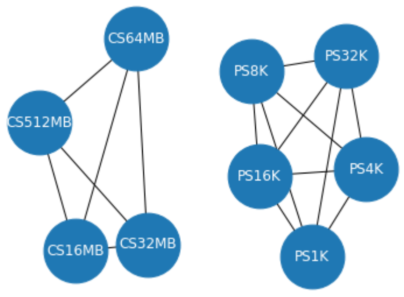

Figure 1.1: Graph of mutually exclusive features in the BerkeleyDB dataset.

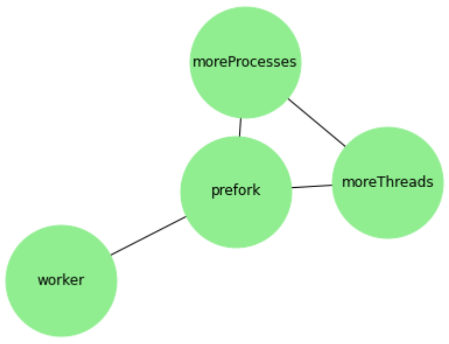

Figure 1.2: Partial graph of mutually exclusive features in the Apache dataset.

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] Leaving 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. So to determine the independent features we just need to determine features, which can be enabled optionally. This can be done by iterating over all features and check 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 group without being concerned of 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 which 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 the amount of components $|C|$ in the mutually exclusive graph. The amount of features in a group can be described with the following formula:

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

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

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

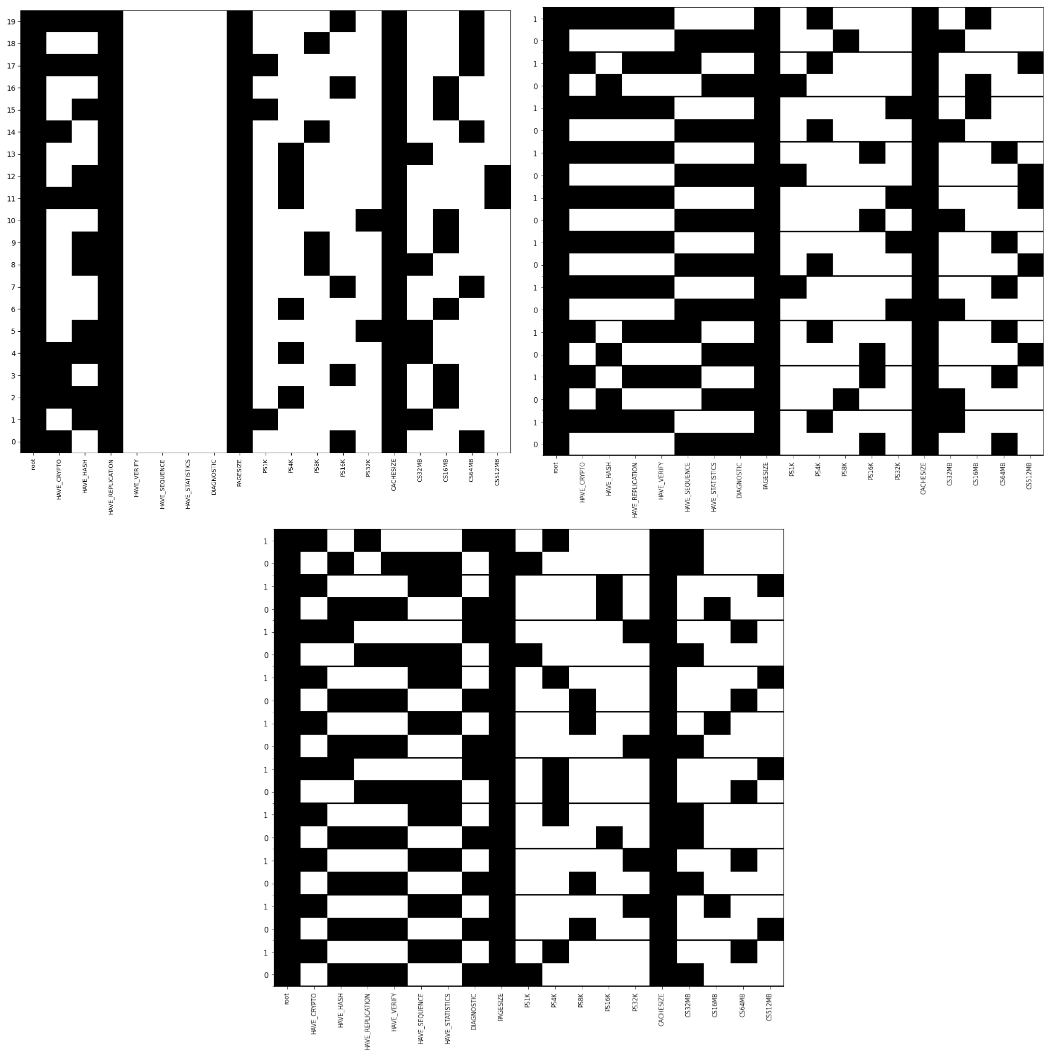

Figure 1.3: Left: Random Sampling - Diversity Promotion - Sample size 20 **Right:** Group Sampling - Hemming Distance - Group Size 5 - Groupings 4

1.5 Influence model

1.5.1 Multicollinearity

- Variance Inflation Factor (VIF)

    – Nicht nötig bei "only optional" , da cliquen und co?

- Rausrechnen
- s. Feedback.pdf

1.5.2 Determining the group influence

1.5.3 Determining the feature influence