Efficient Design of a Complete Rule Search in Sparsely Populated Search Spaces*

Timo Slawinski, Angelika Krone and Peter Krause

December 1999

Abstract

In the field of data–based fuzzy modeling two approaches are predominately applied: Firstly, the optimization of an entire rule base by minimizing the modeling error and secondly to incrementally set up the rule base by individual tested rules. Following the second approach the search space increases exponentially with the number of input variables. On the other hand, due to the limited amount of available data, the search space becomes more and more sparsely populated. Therefore, part of the possible rules are no longer supported by any data set and can be neglected in the search process. The tree–oriented rule search concept, presented in this paper, takes advantage of this fact and leads to a drastically reduced computational effort.

1 Introduction

The applicability of data–based fuzzy modeling methods depends strongly on an acceptable computing time, especially if many linguistic variables and values are considered. However, few of the fuzzy modeling methods, proposed in literature, can be applied to high dimensional problems. Especially, approaches for optimizing the complete rule base are often not practicable for more than a handful of input variables. For more complex problems a suitable approach is to test and rate single rules, in respect to the data. This reduces the problem of finding a good rule base to the much smaller problem of finding individual relevant rules.

The tree–oriented rule search concept, presented in this paper, is integrated in the Fuzzy–ROSA method (FRM) [1, 2, 3, 4]. However it can easily be transferred to other approaches, if the following constraints are fulfilled: The approach must be based on

*This research was sponsored by the Deutsche Forschungsgemeinschaft (DFG), as part of the Collaborative Research Center 'Computational Intelligence' (531) of the University of Dortmund

testing individual rules and be able to handle generalizing (incomplete) rules. These rules need not to consider all the linguistic input variables provided and therefore the length (combination depth) of the premises may differ (Section 2.1).

Here, for simplification logical conjunctions of output variables are not considered. Then the complexity arises predominantly from the number of input variables and their linguistic values. Therefore, in the following, the influence of the output variables is neglected and only the premises are considered.

On the one hand, the number of possible premises increases exponentially with the number of input variables. On the other hand, the number of available data sets is strictly limited in most applications. In this case, the number of data sets per premise decreases drastically if the combination depth is increased. By exceeding a certain combination depth, a large number of the premises are no longer supported by any data set and when combined with any conclusion will not pass the rule test. If further linguistic expressions are added to such a premise in order to generate a more special premise, the derived premise is also not supported by the data. Thus, it needs no longer be considered in the generation process. The tree–oriented rule search concept takes advantage of this fact.

The paper is organized as follows: In order to obtain a more precise impression of the present search space, the number of possible premises is compared to the number of premises that need to be tested on average (Section 2). In Section 3 we describe the tree–oriented search concept in detail. Finally, the efficiency of the tree–oriented search concept is compared to a previous approach by hand of a benchmark problem and a real world application.

2 Search Space Structure

In this section the size of the search space, i.e., the total number of possible premises, is specified by combinatorial calculations. At first the underlying structure of rules is introduced in Section 2.1. As mentioned above, the number of data sets is usually limited. Taking this into account, the maximum number of premises that can be supported by the available data sets is calculated. Finally, for a simple case an estimation is made of the number of hypotheses that must be tested on average.

2.1 Structure of Rules

The fuzzy rules considered here are of the Mamdani type [5, 3]:

$$\text{IF } P \text{ THEN } C \quad \text{with} \quad P = e_1 \wedge \ldots \wedge e_c \quad \text{and} \quad C = e_Y \tag{1}$$

The premise P consists of a conjunction[1] of c linguistic expressions e referring to input variable X_i and the conclusion C is a linguistic expression e_Y referring to the output variable Y. The number of linguistic expressions c is called the combination depth. A linguistic expression e is defined by $X_i = s_{ij}$, where X_i denotes the i-th input variable and s_{ij} the j-th linguistic value for this variable. The S_i linguistic values (sets) s_{ij} are defined by membership functions $\mu_{ij}(x_i)$, where x_i denotes a real (crisp) input value of the variable X_i. A linguistic output expression is defined analogously. The number of linguistic expressions in the premise can be restricted by the maximum combination depth c_{max}, which is usually chosen to be smaller than the number V of input variables. The premise of a rule can also be interpreted as a linguistic input situation. If $c = c_{max} = V$, i.e., each linguistic variable is considered, we speak of a complete input situation and rule, respectively. Otherwise, if $c < c_{max}$, we speak of an incomplete or generalizing input situation and rule, respectively. In the following, the n-th data set is defined by the tuple $d^n = (x_1^n, \ldots, x_V^n, y^n)$ and D is the total number of data sets.

2.1.1 Restrictions

The FRM and similar approaches allows us to introduce the following restrictions, in order to prohibit the generation of undesired or infeasible rules. To preserve maximum flexibility, these restrictions can be activated optionally.

Combination Restriction: The default is to prohibit the combinations of linguistic expressions that refer to the same linguistic variables, because a premise of the type IF "temperature is high" AND "temperature is low" is usually not interpretable.

Complement Restriction: In the FRM each linguistic expression can be used in a negated form to formulate premises of the type if IF "temperature is not high". This restriction is activated as the default, because only few of the common fuzzy tools can process negated statements.

Minimal Data Support: The "trustworthiness" of a rule depends strongly on the number of supporting data sets. In the FRM the (total) data support μ_{sup}^{tot} of a given premise is defined by the sum over the degrees of membership for each data set $\mu_{sup}^{tot} = \sum_{n=1}^{D} \bigwedge_{k=1}^{c} \mu_{i_k j_k}(x_{i_k}^n)$. This restriction allows us to define a minimal data support μ_{sup}^{min}. The default value of the minimal data support is chosen as one. The number of supporting data sets d^s with $\bigwedge_{k=1}^{c} \mu_{i_k j_k}(x_{i_k}^s) > 0$ is denoted with D_{sup}.

[1] In the FRM, the logical AND is realized by the product [6]

2.2 Search Space Analysis and Construction

The size of the search space, i.e. the total number of possible premises P_{tot}^{pos}, increases with the number V of input variable, the number S_i of linguistic values of the variable X_i and the maximum combination depth c_{max}. For the general case, described in [7], it can be complicated to determine P_{tot}^{pos}. Therefore, the following assumptions are made:

- The combination and complement restriction are activated and
- each linguistic variable has the same number of linguistic values (sets) S.

Due to the activated restrictions, each input variable can occur in only one linguistic expression in the premise of the rule. Based on this fact, the number of possible premises P_c^{pos} for a certain combination depth c can be calculated as follows

$$P_c^{pos} = \binom{V}{c} \cdot S^c \quad . \tag{2}$$

The number of possible combinations for the given input variables is calculated by the first factor in Eq. 2 and the second factor considers the influence of the different linguistic values of the input variables. The total number of possible premises P_{tot}^{pos} is the result of a summation over all combination depths

$$P_{tot}^{pos}(c_{max}) = \sum_{c=1}^{c_{max}} P_c^{pos} \quad \text{with} \quad c_{max} \leq V \quad . \tag{3}$$

To compare the total number of possible premises P_{tot}^{pos} with the estimated total number P_{tot}^{sup} of premises that can be supported by the given data sets, the following assumptions are made:

- the partition of the linguistic values is defined as hard, i.e., not overlapping
- the number of data sets is D , and
- the distribution of the data sets is presumed to be the worst case, i.e., the maximum possible number of premises is supported.

The maximum number of premises P_c^{max} of a given combination depth c that can be supported by a single data set is calculated as

$$P_c^{max} = \binom{V}{c} \quad . \tag{4}$$

In comparison to Eq. 2, the number of linguistic values has no influence, because only one linguistic value of each variable is supported by a data set, due to the hard partition.

In the fuzzy case an additional factor 2^c has to be considered in Eq. 4, assuming that a crisp value x_i supports maximal two sets s_{ij_1} and s_{ij_2}. On the other hand, considering a supporting data set d^s the average activation $\bigwedge_{k=1}^{c} \mu_{i_k j_k}(x^s_{i_k})$ of a premise can be roughly estimated by $(1/2)^c$, if normal convex fuzzy sets are assumed, whose membership degrees add up to one in each point. Furthermore, the number D_{sup} of supporting data sets used for the rule test and rating [8, 6], is based on the sum over the activations of the premise (see Section 2.1.1). This two effects compensate each other partly and as shown in our experiments, the hard case is a good approximation.

The maximum number of supported premises P_c^{sup} of the given combination depth c and the total number P_{tot}^{sup} is given by:

$$P_c^{sup} = \min(D \cdot P_c^{max}, P_c^{pos}) \quad \text{and} \quad P_{tot}^{sup} = \sum_{c=1}^{c_{max}} P_c^{sup} \quad , \tag{5}$$

where the number of supported premises P_c^{sup} is limited to the maximum number of possible premises P_c^{pos}. As shown in Fig. 1, for a given combination depth c' the number of supported hypotheses $P_{c'}^{sup}$ is smaller than the number of possible premises $P_{c'}^{pos}$. Applying the tree–oriented search concept presented in Section 3, some of the premises need not be specialized and consequently not all premises of the following combination depth are generated. The number of premises $P_{c=c'+1}^{save}$ not tested is difficult to determine, because it depends on the order in which the premises are generated. Therefore, P_c^{save} and the number of generated premises P_c^{gen} are estimated by calculating the average number of premises not tested

$$\bar{P}_c^{save} = \frac{P_{c-1}^{pos} - P_{c-1}^{sup}}{P_{c-1}^{pos}} \cdot P_c^{pos} \quad \text{and} \quad \bar{P}_c^{gen} = P_c^{pos} - \bar{P}_c^{save} \quad , \quad \text{with} \quad 1 < c' < c \quad . \tag{6}$$

Consequently, the total number of premises P_{tot}^{gen} that must be generated can be estimated by

$$\bar{P}_{tot}^{gen}(c_{max}) = P_{tot}^{pos}(c_{max}) - \bar{P}_{tot}^{save} \quad , \quad \text{with} \quad \bar{P}_{tot}^{save} = \sum_{c=c'+1}^{c_{max}} \bar{P}_c^{save} \quad . \tag{7}$$

The total number of possible premises P_{tot}^{pos} is calculated using Eq. 3 and $\bar{P}_{tot}^{save}$ denotes the estimated total number of premises not tested.

2.2.1 Example

The numbers of possible, supported, saved and generated premises are illustrated in Fig. 1 for the following example: number of input variables $V = 20$, number of linguistic values per input variable $S = 5$, and number of data sets $D = 10000$.

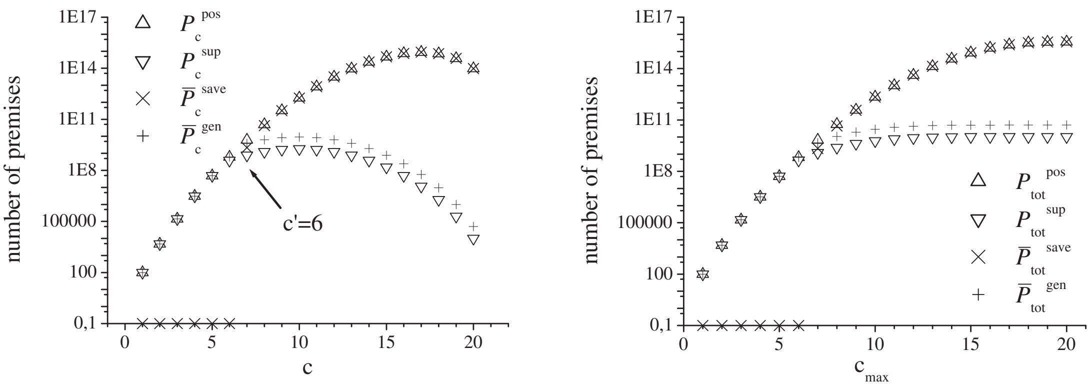

Figure 1: (Total) number of possible, supported, saved, and generated premises as functions of the (maximum) combination depth

Considering the left side of Fig. 1, after the critical combination depth c' is exceeded, it can be seen that the number of generated premises $\bar{P}_c^{gen}$ decreases in a similar manner as the number of supported premises P_c^{sup} with increasing combination depth c. Due to this fact, the increase of the total number of generated premises $\bar{P}_{tot}^{gen}$ is dramatically smaller than the increase of the total number of possible premises P_{tot}^{pos} (right side of Fig. 1).

3 Efficient Complete Rule Search

There are two main approaches for the implementation of a complete rule search. Firstly, all possible premises can be set up and tested (rule–oriented). Secondly, we can set up and test all rules which are supported by the given data sets (data–oriented). In both cases it is guaranteed, that all relevant rules are found, because if no data sets support a rule, latter cannot pass the relevance test.

The computational effort arises mainly from the testing of the rules. For the rule–oriented approach the number of possible premises P_{tot}^{pos} is given by Eq. 3. As discussed above, in the fuzzy case each data set supports $2^c \cdot P_c^{max}$ premises with the combination depth c (compare to Eq. 4). Consequently the possible number P_{tot}^{dat} of premises in the data–oriented approach is given by

$$P_{tot}^{dat} = \sum_{c=1}^{c_{max}} P_c^{dat} \quad \text{with} \quad P_c^{dat} = 2^c \cdot D \cdot P_c^{max} \quad , \tag{8}$$

where P_c^{dat} denotes the number of possible premises of a certain combination depth c in the data–based approach.

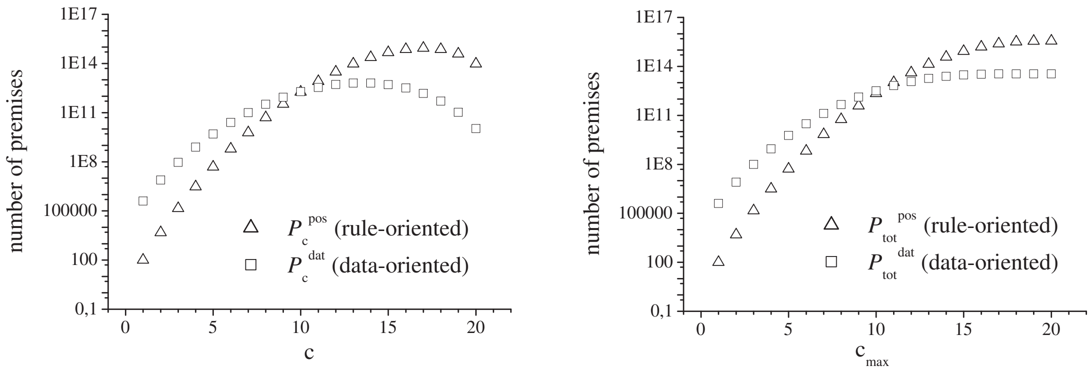

Figure 2: (Total)number of possible premises as function of the (maximum) combination depth for the rule- and data–oriented approach

In Fig. 2 the two approaches are compared for the same example as used in Section 2.2.1. It can be seen that for lower combination depths $c < 10$ the computational effort of the data–oriented approach is higher. For higher combination depths the rule–oriented approach is more time consuming. According to Fig. 1 for combination depth $c > c' = 6$ both approaches test more premises than supported in average after Eq. 7.

It should be noted, that the data–oriented approach can be improved by a mechanism for avoiding repeated testing of premises. On the other hand, such a mechanism can also become very time consuming due to the higher number of premises. Therefore we favorite the tree–oriented search concept presented in the next section.

3.1 Tree–Oriented Complete Search (TOCS)

In the tree–oriented complete search (TOCS), premises are set up and tested with respect to all linguistic output values (rule–oriented). The TOCS concept is based on the linguistic expressions (see Section 2.1). Therefore, the latter are numbered serially: $\tilde{e}_1 = (X_1 = s_{11}, \tilde{e}_2 = (X_1 = s_{12}), \ldots, \tilde{e}_E = (X_V = s_{VS_V})$, where the total number of linguistic expressions is denoted by $E = \sum_{i=1}^{V} S_i$. Starting from the first linguistic expression $\tilde{e}_1$ all more special premises are generated as diagrammed in Fig. 3.

Consider now the n-th linguistic expression $\tilde{e}_n$. For the combination depth $c = 1$ the premise is given by $P = \tilde{e}_n$. Based on this premise, rules are set up by combining it

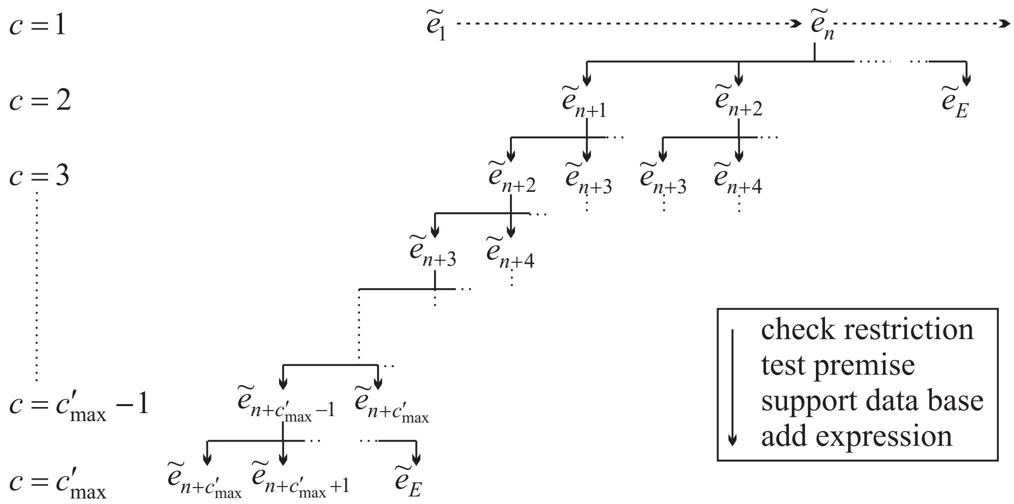

Figure 3: The tree–oriented complete search (TOCS). Starting from a given premise the necessary specialization (generation) steps are indicated in the box. First the restrictions are checked. If the premise is restricted, the branch need not to be considered any more. Then the premise is tested. A result of the test are the supporting data sets, which are taken as a data base in the next specialization step. Finally, a new term is added to the premise.

with all linguistic output values as conclusions. The chosen test and rating strategy is applied, and each rule found relevant is added to the rule base.

Then a new linguistic expression $\tilde{e}_{n+1}$ is added to the premise, until the maximum combination depth c'_{max} is reached. In order to avoid infeasible premises the maximum combination depth is set as $c'_{max} = \min(c_{max}, E-n)$. If the TOCS is applied in this way, all possible rules are set up and tested. By activating combination restriction and minimal data support (Section 2.1), the TOCS can be speeded up drastically as follows:

- A rule is unreasonable in the sense of combination restriction, if two linguistic expressions refer to the same linguistic variable. If this occurs at a certain point, the premise remains restricted, even if it is specialized. Consequently, this branch need not be considered further.
- Rules with a high combination depth are more likely to be not supported by any data sets. Therefore, if a rule is not supported, no more specialized rule with additional linguistic expressions based on that rule can be found relevant. This means that if such a point is reached in a branch of the TOCS, the branch need not be followed further. The computational savings are estimated in Section 2.2.
- Following one branch, a decreasing number of supporting data sets need to be considered. Therefore, the rule test and rating [8, 6] can be implemented very efficiently by considering only the D_{sup} supporting data–sets.

It should be noted that it is also possible to design a tree–oriented search in which rules restricted by the combination restriction are not set up. On the other hand, the TOCS presented here is more flexible and the computational effort to check the restrictions can be neglected in comparison to rule test and rating.

4 Results

In this section the efficiency of the TOCS is studied based on simulations by means of a benchmark problem and a real world application. In the following we give a brief description of the chosen examples and the used hard- and software.

4.1 Examples and Simulation Enviroment

In order to examine the search behavior of the TOCS in different search spaces, we choose two examples with a very different data density, i.e. number of data sets D divided by the total number of possible P_{tot}^{pos}. In the following description the (maximum) number of input variables V, the number of linguistic expressions (sets) S per input variable and the number of data sets D are denoted in brackets, for specifying the search space

wine ($V = 13, S = 5, D = 178$): A classification problem based on data sets[2] that result from chemical analyzes of wine, grown in the same region in Italy, but derived from three different cultivators. The analysis determines the quantities of 13 constituents found in each of the three types of wines. The number S of linguistic values (sets) per variable is five.

load prediction ($V_{max} = 7, S = 7, D = 35040$): The prediction of the electrical load, required by the consumer in a service area, is essential for the operating efficiency and safety of a power control system. Here we consider only the simplified task to predict the load one time step ahead, based on the actual and the previous time steps. The number of variables depends on the maximum time depth $V = t_{max} + 1$. The maximum time depth is given by the maximum combination depth $t_{max} = c_{max} - 1$.

All results are obtained on a Pentium 200 MHz (MMX), 128 MB RAM with the Winrosa 2.0 software tool[3]. For comparison of our results we use the complete search shortened (CSS) [9], which is comparable to the TOCS, if the minimal data support restriction and support data base are not considered.

[2]UCI Repository of Machine Learning Databases
http://www.ics.uci.edu/ mlearn/MLRepository.html

[3]The TOCS is only available in the expert version. More informations and demo software:
http://esr.e-technik.uni-dortmund.de/winrosa/winrosa.htm

4.2 Efficiency of the Tree–Oriented Complete Search (TOCS)

The computational effort arises predominately from the total number of tested premises P_{tot}^{test}. This number[4] can be calculated by $P_{tot}^{test} = P_{tot}^{pos} - P_{tot}^{res}$, where P_{tot}^{res} denotes the total number of restricted premises. As described in Section 4.1 the number of restricted premises P_{tot}^{res} depends on the chosen search method.

In Figure 4 (left) the number of tested premises P_{tot}^{test} for the different search methods is compared to the number of possible premises P_{tot}^{pos} and the estimated number of generated premises P_{tot}^{gen} after Equation 7.

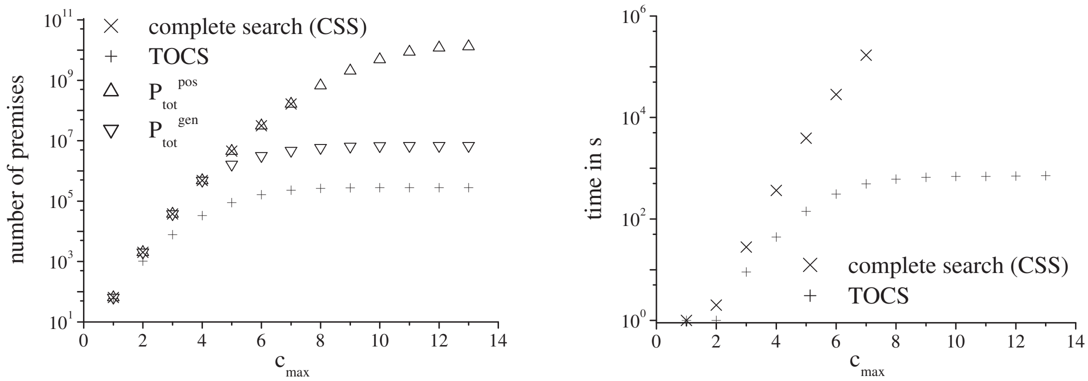

Figure 4: Empirical results for the example wine for the different search methods: In the left chart the number of tested premises is compared to the number of possible premises P_{tot}^{pos} and the estimated number of generated Premises P_{tot}^{gen}. The dependency of the computing time from the maximum combination depth c_{max} is shown in the right chart.

It can be seen that the complete search tests all possible premises. In comparison the number of premises tested by the TOCS is drastically smaller (factor 343 for $c_{max} = 7$). For the same maximum combination depth the generation time for the complete search is 723 times higher than for the TOCS. The factor two for the save of computing time compared to the save of tested premises arises from the support data base, i.e. the rule test in the TOCS need not to consider all data sets for $c_{max} > 2$ (compare to Section 3).

Furthermore it can be seen that the our estimation for the generated premises P_{tot}^{gen} is conservative in the sense that the actual number of tested premise P_{tot}^{test} is always smaller. A possible explanation is, that the worst case for data distribution is underlaid

[4]The Winrosa 2.0 software tool indicates the total number of generated rules R^{gen} and the total number of inadmissible rules R^{res}. The corresponding number of tested premises can be calculated as follows: $P_{tot}^{test} = (R^{gen} - R^{res})/S_Y$, where S_Y denotes the number of linguistic values (sets) of the output variable. Thereby we assume, that only one output variable is considered and consequently each premise has to be tested in respect to all linguistic output values.

for the calculation of P_{tot}^{gen} in Section 2.2. However, the estimation and the simulation show a similiar behavior. First, the number of testet/generated premises increases exponentially. Then, if a certain maximum combination depth is exceeded the number of tested/generated premises and also the computing time is nearly constant. Consequently, if this critical point can be reached it is possible to find all relevant rules with a maximum combination depth up to $c_{max} = V$.

Considering now Figure 4, in this example this critical point is not reached yet, due to the much larger amount of data sets. Nevertheless the save of computing time is factor 739 for a maximum combination depth $c_{max} = 7$, even though the save of tested premises is only factor nine. In this example the save of computing time for the lower combinations depths is predominately caused by the decreasing number of support data sets D_{sup} (compare to Section 3).

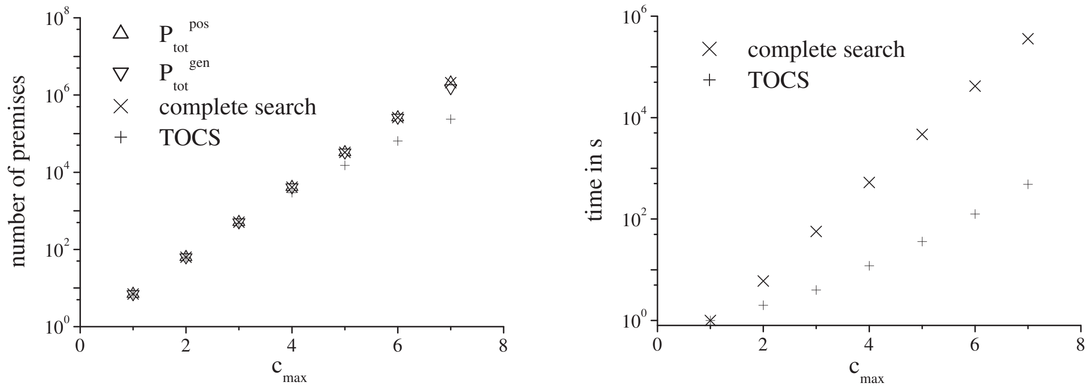

Figure 5: Empirical results for the example load for the different search methods: In the left chart the number of tested premises is compared to the number of possible premises P_{tot}^{pos} and the estimated number of generated Premises P_{tot}^{gen}. The dependency of computing time from the maximum combination depth c_{max} is shown in the right chart. Different from the example wine, here the number of variables $V = c_{max}$ depends on the maximum combination depth.

4.3 Influence of the Minimal Data Support

For the results presented so far the default value $\mu_{sup}^{min} = 1$ has been chosen for the minimal data support restriction. By increasing it, more premises are restricted and this leads to a smaller search space, i.e. less premises have to be generated and tested. However, if a critical value for the minimal data support is exceeded, not all relevant rules are found.

In Figure 4 (left) the number of tested premises P_{tot}^{test} applying the TOCS is compared to the estimated number of generated premises P_{tot}^{gen} after Equation 7 for different values

μ_{sup}^{min} of the minimal data support. It can be seen that the search space for $\mu_{sup}^{min} = 8$ is 26 times smaller than for $\mu_{sup}^{min} = 1$. Also the computing time in the same case is still eleven times smaller (right chart of Figure 4). For higher values of the minimal data support not all relevant rules are found, (e.g. 98.4 % for $\mu_{sup}^{min} = 9$). It has to be noted, that the critical value depends on the problem by hand and the chosen rule test [8].

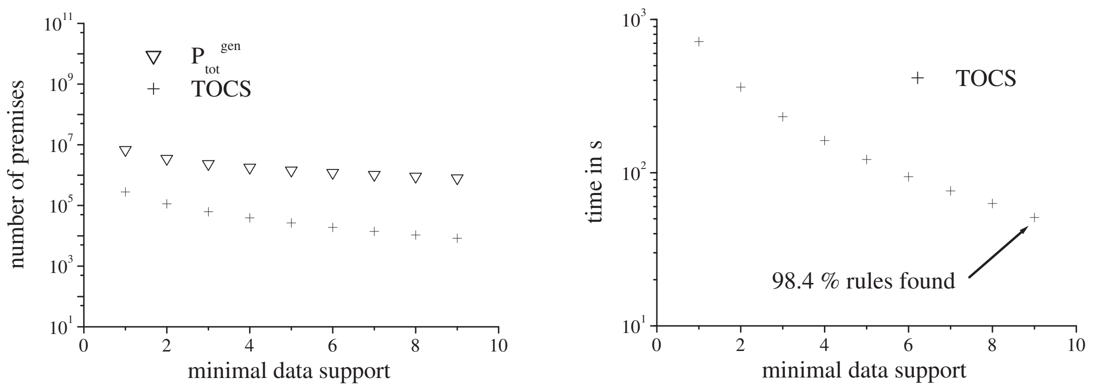

Figure 6: Empirical results for the example wine depending on the minimal data support μ_{sup}^{min}: In the left chart the number of premises tested by the TOCS is compared to the estimated number of generated Premises P_{tot}^{gen}. The dependency of the computing time from the minimal data support μ_{sup}^{min} is shown in the right chart.

5 Summary

The tree–oriented complete search (TOCS) presented in this paper leads to a drastically reduced computational effort in sparsely populated search spaces. This is due to the fact, that not all possible rules of a certain combination depth are supported by data sets and therefore need not to be considered further in the search process. An estimation is made based on search space structure analysis for the number rules, which need not to be tested by taking advantage of this fact. Our simulations validate the theoretical results, by hands of a benchmark problem and a real–world application.

References

[1] A. Krone and H. Kiendl. Automatic generation of positive and negative rules for two–way fuzzy controllers. In *Proceedings of the Second European Congress on Intelligent Techniques and Soft Computing , EUFIT '94*, volume 1, pages 438–447, Aachen, 1994.

[2] A. Krone and H. Kiendl. Evolutionary concept for generating relevant fuzzy rules from data. *International Journal of Knowledge–based Intelligent Engineering Systems*, 1(4):207–213, 1997.

[3] H. Kiendl. Computing with words in Information/Intelligent systems. chapter Decision Analysis by Advanced Fuzzy Systems, pages 223–242. Physica-Verlag Heidelberg, 1999.

[4] T. Slawinski, A. Krone, U. Hammel, D. Wiesmann, and P. Krause. A hybrid evolutionary search concept for data-based generation of relevant fuzzy rules in high dimensional spaces. In *Proceedings of IEEE International Conference on Fuzzy Systems, FUZZ–IEEE '99*, volume 3, pages 1432–1437, Seoul, Korea, 1999.

[5] E. H. Mamdani and B. R. Gaines. *Fuzzy Reasoning and its Applications.* Academic Press, London, 1981.

[6] A. Krone and H. Taeger. Relevance test for fuzzy rules. In *Reihe Computational Intelligence.* CI–40/98, Sonderforschungsbereich 531, Universität Dortmund, 1998.

[7] A. Krone, T. Slawinski, and P. Krause. Search space structuring as a key to cope with different problem sizes in the field of fuzzy modeling. In *Proceedings of World Automation Congress, WAC '00*, Maui, Hawaii, USA, 2000.

[8] H. Jessen and T. Slawinski. Test- and rating strategies for data–based rule generation. In *Reihe Computational Intelligence.* CI–39/98, Sonderforschungsbereich 531, Universität Dortmund, 1998.

[9] A. Krone and H. Kiendl. *WINROSA From Data to Rules.* MIT-Managment Intelligenter Technologien GmbH, Aachen, 1997. Manual.