Preprocessing Data for Machine-Learning Algorithms to Provide User Guidance in Special Purpose Machines

Sascha Lang*, Valentin Plenk†
Institute of Information Systems at Hof University
Hof, Germany
Email: *sascha.lang@iisys.de, †valentin.plenk@iisys.de

***Abstract*—In our previous work, we proposed a system which makes complex production machines more user-friendly by giving the recommendations to the operator. So, we assist the user working with a complex production machine. The recommendations are presented like: "In the last 10 occurrences of this event the operators performed the following keystrokes". While working on the project, we had problems with retrieving the correct recommendations from our knowledge base. Meanwhile, we gathered more data from our project partners. Now, we dive deeper into this data in order to improve our solutions. This work describes methods to preprocess the data. This preprocessing should help us building up the knowledge base. To achieve this automatically, we do not want to know much about the machine and the production process itself.**

Keywords–machine-learning; human machine interfaces; special-purpose machines; production machines; case based reasoning

I. INTRODUCTION

In [1], we used machine learning algorithms to generate recommendations like "in the last 10 occurrences of this event the operators performed the following keystrokes" for operators of complex production machines. Figure 1 shows the basic structure of a plastic extruder. Figure 2 draws the basic structure of our system. The left side represents the machine, in our case, the extruder. The right side shows the structure of our system. Its purpose is to build up a knowledge base from operator interactions which were performed in the past. Since, we started our project, the surroundings stayed the same, as discussed in [2].

Our system is faced with two challenges: First is retrieving the recommendation that is most suitable for the current state of the production machine. Second is building up a knowledge base by extracting the operator interactions from data gathered from the production machine.

In our previous paper [3], we evaluated our existing algorithms, which were created to retrieve recommendations from our knowledge base. But because, we were not satisfied with the results, we decided to take one step back. Instead of improving our retrieval algorithms, we now focus on the way we create the knowledge base from the data, we got from our project partners. We developed some improvement steps, and we will check how they influence our retrieval results .

A similar work is done by N. Ben Rabah et. al. [4]. They also create their knowledge base automatically. In contrast to our project they use two knowledge bases: One for normal states and one for failure states. We on the other hand use only one knowledge base. This solely stores the situations, in which the machine needs attention. Another difference is that their data only consists of binary values. We are faced with all kinds of numerical data. Beside bit values, we also deal with floating point numbers of different ranges.

Some other approaches, like the "APPsist" system [5] or the system described by Cen Nan et.al. [6] have a manually build up knowledge base. We on the other hand focus on creating the knowledge base autonomous. So, we are faced with the challenge to recognize important subsets of our gathered data automatically, which means without the help of a process expert.

Before providing a possible solution, we have to define some datasets. Therefore, we created three data sets from different machines. All machines are plastic extruders. The ones used for Dataset 5 and Dataset 6 contain the same control system. The machine for Dataset 7 uses a different type of

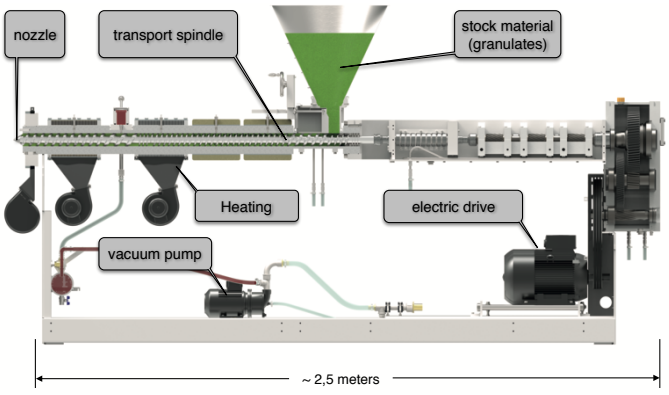


Figure 1. Plastics extruder
(courtesy: Hans Weber Maschinenfabrik, Kronach, Germany)

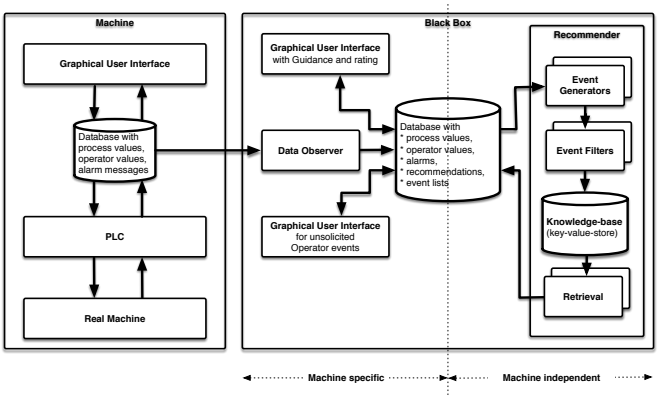


Figure 2. System architecture

 ISBN: 978-1-61208-670-5

control system which contains no historical data. The period how long each dataset lasts, was registered. We also logged the time for which the machine was productive during that period. These information can be seen in Table I. The biggest difference we found between the datasets relates to the amount of user operations we found. Dataset 7 has far the least amount of operations but nearly the longest runtime.

TABLE I. DESCRIPTION OF THE TEST DATASETSS

	Dataset		
	5	**6**	**7**
Start	17-11-05	17-11-05	17-11-26
End	18-03-05	18-04-23	18-04-23
Runtime	$\approx 758h$	$\approx 1664h$	$\approx 1645h$
Events	1159	537	147

The data, we get from our productions machine are stored in a database table, an example is shown in Table II. Each column, contains the value either for a process value or a operator value. We get this data every 10 seconds. The tables for Dataset 6 and 7 are similar, but have a different amount of columns. In our data, we have numerical values as for temperature and rotational speed as well as percentage and bit values.

The next section describes the problems, we discovered while working on our prior approach. Section III describes how, we process the data from the machine to create the values for the knowledge base. In Section IV, we describe our methods to get better data for the keys. The last Section V summarizes the results and gives a short description of the problems, we have to solve in the future.

II. Problems with prior approach

In our prior approach, the trigger point for an entry in our knowledge base was the beginning of an alarm situation. For this, we needed an additional table containing the start and end times of every alarm. For every alarm, we generated a fingerprint and an operator sequence. This pair represents a key (fingerprint) value (operator sequence) entry in the knowledge base. If a new alarm is raised, we will create the respective fingerprint for it. Now, we search for the most similar key inside the knowledge base. The value belonging to this key is then used to present a recommendation to the operator.

A. Problems with alarms

One problem, we had with our alarm based approach is that, we have to deal with simultaneous alarms. For example, if a temperature alarm for a specific zone is raised, it would also raise general temperature alarm additonaly.

In our data, we discovered three types of simultaneous alarms. The first type has the exact same start and end timestamp, so therefore it lasts for the exact same time period and is simply treated as one alarm. The second group are nested alarms. In this case one alarm is raised while another alarm is active. And it is finished before the first alarm is finished. The last group are so called staggered alarms. In this case a second alarm is raise while another one is still active. But in contrast to the nested alarms it is finished after the first alarm.

Alarms lasting the exact same time period, are treated one alarm. Now let s be the start of an alarm and e its end and

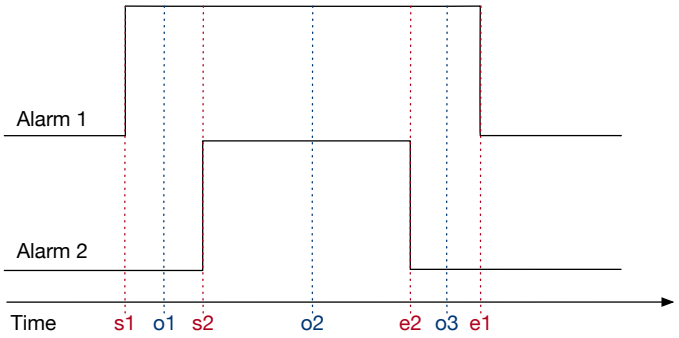


Figure 3. Nested Alarms

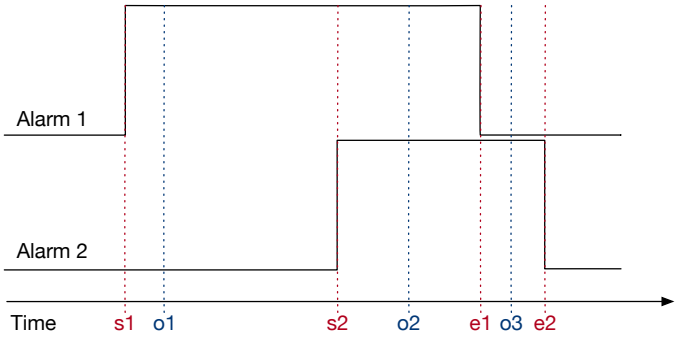


Figure 4. Staggered Alarms

be o some operator events occured during the alarm situation. According to Figure 3 and Figure 4, we were faced with the following situations:

- If $o1$ and $o3$ occur, we can say that the ne alarm should last from $s1$ to $s2$.
- If only $o2$ occurs or $o2$ and $o3$ occur, it is not sure if $s1$ or s2 should be the start of the resulting alarm.
- If only $o1$ occurs or $o1$ and $o2$ occur, it is not sure if $e1$ or $e2$ should be the end of the resulting alarm.
- If only $o2$ occurs, it is not sure which start and end point would be correct for the remaining alarm.

B. User interaction beside of alarms

While analyzing the data more intense, we discovered another problem. We noticed that, we had some user interactions beside the alarm situations. So if, we only consider the situations, with an alarm being raised, we will not consider all situations, in which it will be necessary to change settings of the machines. But because, we want to support the user during the whole time, we have to consider all operator events not only the ones raised during an alarm.

C. Problem with similarity of fingerprints

By solving these problems, we created a knowledge base for every dataset. But now, we noticed some issues with our fingerprints. We expected that entries with the same value (operator-sequence) would have similar keys. But this expectation was not satisfied, we have to figure out how, we can solve this problem.

 ISBN: 978-1-61208-670-5

TABLE II. EXAMPLE DATA FROM DATASET 5

Time	Extruder.Temp0.PV	Extruder.Temp0.OP	Extruder.Temp0.SP	Meltpump.Out.Val	Production_Mode
17-11-05 00:00:06	200.0	34.0	200.0	26.400	1
17-11-05 00:00:16	200.1	34.0	200.0	26.400	1
17-11-05 00:00:26	200.1	30.2	200.0	26.400	1
17-11-05 00:00:36	200.1	33.3	200.0	26.400	1

TABLE III. LIST OF DELAYS BETWEEN EVENTS IN DATA SET 5

Delay(s)	Count	Delay(s)	Count
$d \leq 10$	823	$10 < d \leq 60$	89
$60 < d \leq 160$	17	$160 < d \leq 300$	17
$300 < d \leq 600$	37	$600 < d \leq 1200$	36
$1200 < d \leq 2400$	18	$2400 < d \leq 3600$	8
$3600 < d \leq 7200$	26	$7200 < d \leq 10800$	18
$10800 < d \leq 14400$	6	$14400 < d \leq 18000$	3
$18000 < d \leq 36000$	17	$36000 < d$	43

TABLE IV. LIST OF DELAYS BETWEEN EVENTS IN DATA SET 6

Delay(s)	Count	Delay(s)	Count
$d \leq 10$	128	$10 < d \leq 60$	37
$60 < d \leq 160$	13	$160 < d \leq 300$	8
$300 < d \leq 600$	12	$600 < d \leq 1200$	13
$1200 < d \leq 2400$	24	$2400 < d \leq 3600$	18
$3600 < d \leq 7200$	25	$7200 < d \leq 10800$	14
$10800 < d \leq 14400$	5	$14400 < d \leq 18000$	7
$18000 < d \leq 36000$	29	$36000 < d$	72

III. GENERATION OF OPERATOR SEQUENCES

To solve the problems described in Section II, we decided to not consider alarms from now on. Because, we are able to extract operator events from the machine data, we will use this events as a trigger point. Around these trigger points, we will generate some kind of an own ”alarm”.

To create operator sequences to the respective the events, we have to decide how long the pause between two events should be before they are considered as different sequences.

A. Using the Hauloff Speed

Our first idea to determine this pause is, to evaluate, how much time an operator needs to make changes to the machine. So, we interviewed some machine operators. These conversations revealed that the time depends on the product. How fast a product is created depends on the so called Haul Off Speed. Together with an information of the extruder length, we can estimate the time of the intermission.

Unfortunately none of our test systems provides the Haul Off Speed. So, we talked to a process expert to give us an estimation for a rough maximum and minimum value. In our case the main products are pipes, so the expert mentioned that the haul off speeds will be between $1\frac{m}{min}$ and $50\frac{m}{min}$. If, we have an extruder being, e.g., 35 m long, the resulting period is between $\approx 45s$ and $\approx 2100s \approx 35min$.

B. Using a constant ratio

For this method, we create a list of all pauses between one event and its subsequent event. This list is ordered from small to long delays. We now assume that, e.g., the biggest 20 percent of this delays are pauses between two sequences. The event counts for Dataset 5 is shown in Table III, for Dataset 6 in Table IV, for Dataset 7 in Table V.

TABLE V. LIST OF DELAYS BETWEEN EVENTS IN DATA SET 7

Delay(s)	Count	Delay(s)	Count
$d \leq 10$	27	$10 < d \leq 60$	32
$60 < d \leq 160$	10	$160 < d \leq 300$	11
$300 < d \leq 600$	3	$600 < d \leq 1200$	8
$1200 < d \leq 2400$	5	$2400 < d \leq 3600$	2
$3600 < d \leq 7200$	4	$7200 < d \leq 10800$	2
$10800 < d \leq 14400$	0	$14400 < d \leq 18000$	0
$18000 < d \leq 36000$	2	$36000 < d$	29

According to this table, we found 1115 pauses and therefore 1116 events. According to our 80/20 rule, we say that the lowest time delays are too short to determine a intermission between two operator sequences. So all time delays, which are longer or equal than160 s separate two operator sequences. Because in the other Datasets using this ratio exceeds the boundaries, we determined in the last Section, we also used a ratio of 50/50, which resulted in a pause of 10 s.

C. Elbow method

For this method, we start with a pause of 10 s. We subsequently increase this by another 10 s. Now the sequences are generated for every step and then they are counted. The count of sequences is plot, as shown in Figure 5. We now search for a horizontal area within the boundaries defined in Section III-A. We use either the first one, which stays ten times horizontal or the longest within the boundaries. The start of the resulting horizontal area is the resulting pause.

D. Comparison of the methods and Discussion

Now, we perform the methods described above on all our Datasets. The results are shown in Table VI. For all but one datasets the 80:20 ratio resulted in a bigger value than the upper bound, we determined. From all of the results, which are in between the boundaries, we choose the smallest number further on. We will use this further on for this work.

TABLE VI. COMPARISON BETWEEN THE METHODS TO DETERMINE THE LENGTH OF THE OPERATOR SEQUENCES

Method	Dataset		
	5	6	7
80:20	160	26322	58807
50:50	10	290	162
Ellbow	1160	950	420
Choosen value	160	290	162
Resulting Sequences	162	203	59

IV. SIMILARITY OF FINGERPRINTS

After, we have determined how operator sequences can be created, we will improve our fingerprint data. In a first step, we grouped the fingerprints by their respective operator sequence. Now, we expected that the fingerprints inside a group are similar to each other, and to the fingerprints of other

 ISBN: 978-1-61208-670-5

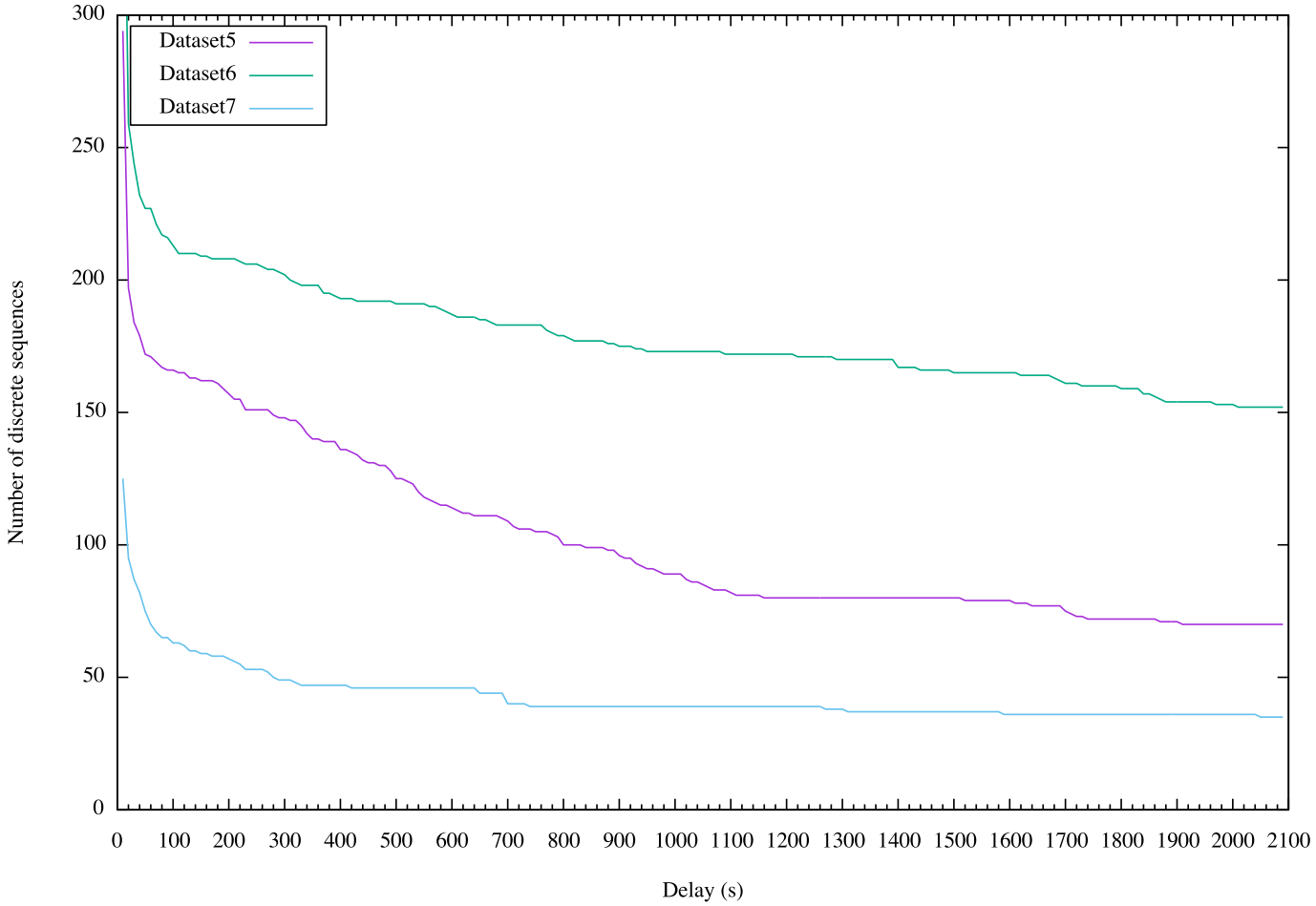


Figure 5. Results for the elbow method

groups they are dissimilar. So our target is to get our data more selective. This will be done by choosing the most useful distance measure for each dataset, then filtering the important columns and finally testing different fingerprint lengths. The results of these steps will be printed afterwards.

A. Preparing the data

Before, we start improving our data, we do some preparing steps. For example, we sort out some unused or cumulative columns. And we perform normalization because of the different ranges of the columns so every one is in the range between 0 and 1.

1) Determining unnecessary columns: As, we investigated our data further, we found three types of columns, which seem not to be helpful and therefore, should not be used further on:

- Columns, which are not used, e.g., having a null value
- Columns having only one value over the whole time
- Cumulating columns, e.g., we have a column counting the energy the machine needed

2) Normalization of columns: The next issue, we discovered is that ranges, e.g.:

- Temperate ranges from $\approx 0°C$ to $\approx 250°C$
- Percentage values (OP), from 0 to 100
- Pressure values from 0 to 60
- Button or Binary values from 0 to 1

Using this normalization, we get a range between 0 and 1 for all columns (Equation 1).

$$x_i' = \frac{x_i - mean(\vec{x})}{max(\vec{x}) - min(\vec{x})} \tag{1}$$

B. Optimization of the Fingerprints

One of our major goals is to achieve the improvement of the fingerprints autonomous. To achieve this, we have to define a quality-measurement. For every case, we have a pair of a key (fingerprint) and value (operator sequence). According to the values they can be divided in the following cases:

- both pairs have the same operator sequence: d_{int}
- both pairs have different operator sequences: d_{ext}

Now, we assume that a good fingerprint will have a small d_{int} and a big d_{ext}. Let i be the number of results for d_{int} and e the number of results for d_{ext}, we can now define two ways to express our quality measure:

$$d_{sum} = \left(\sum_{n=1}^{e} d_{ext}\right) - \left(\sum_{n=1}^{i} d_{int}\right) \tag{2}$$

 ISBN: 978-1-61208-670-5

TABLE VII. AVAILABLE DISTANCE MEASUREMENTS

Point set distances	Point distances
Hausdorff distance	Manhattan distance
Average Linkage	Chebychev distance
Single Linkage	Euclidean
Maximum Linkage	Squared Euclidean

and

$$d_{avg} = \frac{(\sum_{n=1}^{e} d_{ext})}{e} - \frac{\left(\sum_{n=1}^{i} d_{int}\right)}{i} \quad (3)$$

For now on, we will use the d_{avg} (Equation 3).

1) Choosing the most suitable Distance Measurement: Now, we want to determine the best working distance measurement. As described in our previous work [1], we use the distance measurements as listet in Table VII.

For every possible combination of point set and point distances, we determine the quality. As the different distance measures will result in different value ranges, we also have to normalize this results. The combination resulting in the highest quality will be used. The results for all three Datasets are shown in Table VIII.

2) Filtering the Columns: After, we have determined a suitable distance measurement, we are faced with the question if we only have relevant columns in our data. We assume, that using all columns will lead to a worse result than using only some.

At first, we reduce the fingerprint to one column. For this column d_{avg} is determined. This step is repeated for all other columns in the data.

The next step is a Greedy like algorithm. We sort the results from the last step. The column with the highest value is set for the resulting fingerprint. Now, we subsequently add the next columns and again determine d_{avg}. If the new quality is at least 5% better than the old one, the column will be added to the fingerprint.

After doing this for all columns, we finally get a set of columns for our resulting fingerprint. Beside the columns found in this algorithm the fingerprint also contains the columns containing data regarding operator events and columns, we need to determine wether the machine is manufacturing the product or not. As for the last step the results for this improvement step are also shown in Table VIII.

3) Fingerprint Length: In our last papers, we set the fingerprint length to exactly 5 minutes. We always struggled with this decision and wanted to verify, if this would be a suitable number. We now want to determine a suitable length for the fingerprint. At a first step, we should specify a upper and a lower bound. The lower bound is the smallest possible length of a fingerprint. As, we get our data every 10 seconds, the smallest possible fingerprint is 10 seconds long. The upper bound will be set to 30 minutes for now. The quality for every range is determined. The timespan with the best quality will be used. The results for this step are also shown in Table VIII.

C. Results

Table VIII shows the results for the improving steps. Now the matrix plots shown in Figures 6, 7, 8 are created for every dataset and every improvement step to show how our improvement has worked out. The color of each cell shows the similarity between two fingerprints. A green color shows a close distance, the red color a far distance. The operator sequences are grouped. So inside a white rectangle they all fingerprints have the same operator sequence. The perfect result would be green groups lying on the diagonale and red groups elsewhere. An Additional matrix plots, ordered by time is added.

According to the results in Table VIII for two Datasets, a short fingerprint was a better choice. The distance measurement is also the same for every Dataset.

1) Results on Dataset 5: Table IX lists all operator sequence which, we have found in Data set 5. The Shortcuts are used as labels in Figure 6. We have found overall 125 sequences with 21 different operator sequences. The step with the most difference seem to be the step using the Chebychev distance instead of the Euclidean distance. Also the step shorting the fingerprint makes a noticeable difference. Overall, we seem to get more red squares beside the diagonal line, this seems to be a step in the right direction. We are not satisfied with this result at all, because, we do not have green rectangles on the diagonale, as, we hoped

2) Results on Dataset 6: In Table X lists all operator sequence which, we have found in Data set 6. The Shortcuts are used as labels in Figure 7. We have found overall 145 sequences with 14 different operator sequences. As in Dataset 5 the step with the most difference seem to be the step using the Chebychev distance instead of the Euclidean distance. In this dataset, we noticed that a bigger fingerprint performs better, but, we do not see such a noticeable difference in the result as in Dataset 5. With this results, we are also not satisfied yet.

3) Results on Dataset 7: In Table XI lists all operator sequence which, we have found in Data set 7. The Shortcuts are used as labels in Figure 8. We have found overall 21 sequences with nine different operator sequences. Dataset 7 has the most interesting results. Unfortunately it is also the smallest data set, regarding to the number of operator sequences, we found. Changing the distance to Chebychev resulted in a green diagonale and now only the elements lying on this diagonale seem to be similar. The next step, filtering the columns anyway seems to improve the result a bit. The final step, changing the fingerprint length, also changed the result. We are not sure if, this is an improvement or not. Although this Dataset let us hope that, we can achieve some improvement of our data with the algorithms described in this work.

V. Conclusion and Future Work

As described in Section IV, we tried to improve our results by optimizing the data, we gathered from the production machines. For nearly all of our improvement steps, we found a more or less noticeable difference in the results. The most distinct change was for changing the distance measurement. In this case the point set distance stayed at the Hausdorff distance, but the point distance was changed from the Euclidean to Chebychev distance. We have a lot of work to do, because our results are yet not satisfying. As we noticed some changes, especially regarding to Dataset 7, we think we are on the course to get some improvements in the future.

By now the sequence in which, we perform the improvement steps is arbitrarily chosen. By changing this order or

TABLE VIII. RESULTS OF THE OPTIMIZATION ALGORITHMS

Optimizations	Dataset 5	Dataset 6	Dataset 7
Columns original	120	225	177
Useless columns	47	75	56
Point Set Distance	Hausdorff	Hausdorff	Hausdorff
PointDistance	Chebychev	Chebychev	Chebychev
Columns	Tool.Temp2.OP Extruder.Temp0.OP Extruder.Temp4.OP Vacuumtank.In.Vacuum1.Val Meltpump.In.RPM.Band_Min Vacuumtank.In.Vacuum0.Val Tool.Temp0.OP	General.In.Power.Band_Min Extruder.In.Power.Band_Min General.In.Power.Val Vacuumtank.In.WaterTemperature3.Band_Max Vacuumtank.In.WaterTemperature1.Val Tool.Temp9.PV Tool.Temp19.PV Tool.Temp5.OP Tool.Temp18.SP	N2_PDO_Istwert1[1] N2_PDO_Istwert4[1] PDO5_PLCAnalog_In11_12[0] N1_PDO_Istwert3[4] N1_PDO_Istwert4[4]
Length (s)	70	1850	30

TABLE IX. OPERATOR SEQUENCES FOUND IN DATA SET 5

Shortcut	Sequence	Count
S1	Do nothing	2
S2	Meltpump.Out.Val_-0.1#	9
S3	Meltpump.Out.Val_-0.2#	8
S4	Meltpump.Out.Val_-0.3#	7
S5	Meltpump.Out.Val_-0.4#	4
S6	Meltpump.Out.Val_-0.5#	4
S7	Meltpump.Out.Val_-0.6#	2
S8	Meltpump.Out.Val_-0.7#	2
S9	Meltpump.Out.Val_-0.8#	2
S10	Meltpump.Out.Val_-1.0#	4
S11	Meltpump.Out.Val_0.1#	16
S12	Meltpump.Out.Val_0.2#	23
S13	Meltpump.Out.Val_0.3#	12
S14	Meltpump.Out.Val_0.4#	7
S15	Meltpump.Out.Val_0.5#	6
S16	Meltpump.Out.Val_0.6#	5
S17	Meltpump.Out.Val_1.0#	4
S18	Meltpump.Out.Val_1.7#	2
S19	Meltpump.Out.Val_2.0#	4
S20	Meltpump.Out.Val_6.9#	2
S21	Tool.Temp3.SP_10.0#	2

TABLE X. OPERATOR SEQUENCES FOUND IN DATA SET 6

Shortcut	Sequence	Count
S1	Do nothing	2
S2	Meltpump.Out.Val_-0.1#	20
S3	Meltpump.Out.Val_-0.2#	21
S4	Meltpump.Out.Val_-0.3#	4
S5	Meltpump.Out.Val_-0.4#	7
S6	Meltpump.Out.Val_-0.5#	4
S7	Meltpump.Out.Val_-0.6#	4
S8	Meltpump.Out.Val_-0.8#	2
S9	Meltpump.Out.Val_0.1#	17
S10	Meltpump.Out.Val_0.2#	28
S11	Meltpump.Out.Val_0.3#	21
S12	Meltpump.Out.Val_0.4#	12
S13	Meltpump.Out.Val_0.5#	2
S14	Meltpump.Out.Val_0.6#	3

TABLE XI. OPERATOR SEQUENCES FOUND IN DATA SET 7

Shortcut	Sequence Count	
S1	Do nothing	4
S2	N1_SP0_10.0#N1_SP1_10.0# N1_SP2_10.0#N1_SP3_10.0#	2
S3	N1_SP0_20.0#N1_SP1_20.0# N1_SP2_20.0#N1_SP3_20.0#	2
S4	Analog_Out_-2.0#	2
S5	Analog_Out_-8.0#	2
S6	Analog_Out_1.0#	2
S7	Analog_Out_10.0#	2
S8	Analog_Out_20.0#	3
S9	Analog_Out_7.0#	2

repeating some of the steps we might get a further improvement. E.g., if we determine the fingerprint length first, then perform column optimization and finally improve the fingerprint length again. Also, we should run the column filtering with all possible distance measurements combinations. This might result in a better set of columns then using only one.

In Section IV, we also mentioned that, we use 5% as a threshold to detect whether a improvement in quality is significant or not. We should question this value and evaluate if a smaller or bigger threshold can improve the results. Also, we currently use the operator event data for both, fingerprint and operator sequence. We should also verify if it is better to use them only if they are detected by our improvement algorithm.

To verify the result, we can, e.g., use the method, we developed in our previous work [1] together with the matrix plots, we should be able to determine whether, we get better results or not.

Another problem, we have with some improvement steps is the long computing time. So, we have to discover how much data, we have to have to get a proper improvement. Or how often the system should run the steps (e.g., once a week or once a month). After performing the improvement steps, the system has to create the knowledge base again, so, we should keep this in mind.

The retrieval methods we currently use to get recommendations out of our knowledge base, do not consider the order of the data points. In our future work, we will also consider other distance measurements, which take care about the order of the points. One of our current ideas is taking the data no longer as a n-dimensional point-set but consider it as a n-dimensional polygonal chain.

An advantage of the described algorithms is that they do not need any information about the process. So therefore they are not influenced by the operator and can easily be adopted to other process types. This is important because the transferr our system to other processes will be a major part of our future work.

REFERENCES

[1] V. Plenk, S. Lang, and F. Wogenstein, “An approach to provide user guidance in special purpose machines and its evaluation,” International Journal On Advances in Software, vol. 10, no. 3 and 4, December 2017, pp. 167–179.

 ISBN: 978-1-61208-670-5

[2] V. Plenk, “Improving special purpose machine user-interfaces by machine-learning algorithms,” Proceedings of CENTRIC 2016: The Ninth International Conference on Advanves in Human-oriented and Personalized Mechanisms, Technologies and Services, August 2016, pp. 24–28.

[3] V. Plenk, S. Lang, and F. Wogenstein, “Scoring of machine-learning algorithms for providing user guidance in special purpose machines,” October 2017, pp. 7–13.

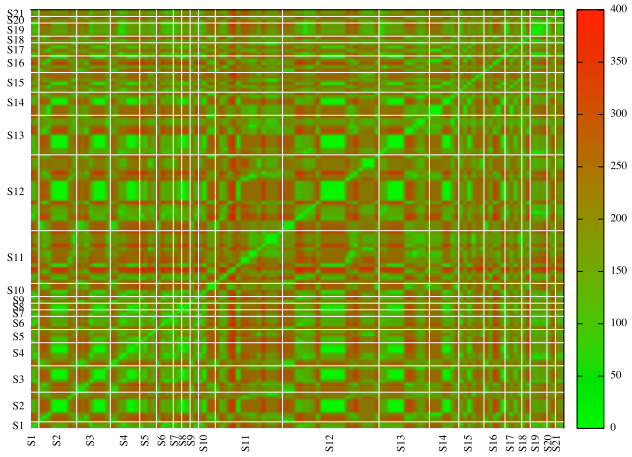
[4] N. Ben Rabah, R. Saddem, F. Ben Hmida, C.-M. V., and M. Tagina, “Intelligent case based decision support system for online diagnosis of automated production system,” 13th European Workshop on Advanced Control and Diagnosis (ACD 2016), 2016.

[5] C. Ullrich, M. Aust, M. Dietrich, N. Herbig, C. Igel, N. Kreggenfeld, C. Prinz, F. Raber, S. Schwantzer, and F. Sulzmann, “Appsist statusbericht: Realisierung einer plattform für assistenz-und wissensdienste für die industrie 4.0.” in DeLFI Workshops, 2016, pp. 174–180.

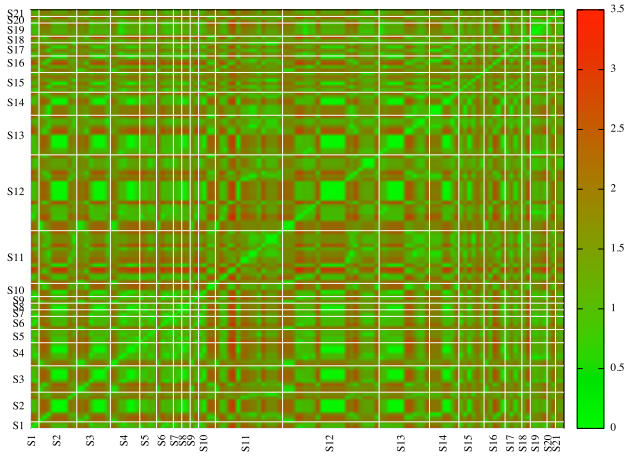
[6] N. Cen, K. Faisal, and I. M. Tariq, “Real-time fault diagnosis using knowledge-based expert system,” Process Safety and Environmental Protection 86, no. 86, 2007, pp. 55–71.

[7] J. Lunze, Ereignisdiskrete Systeme. München: Oldenbourg, 2006.

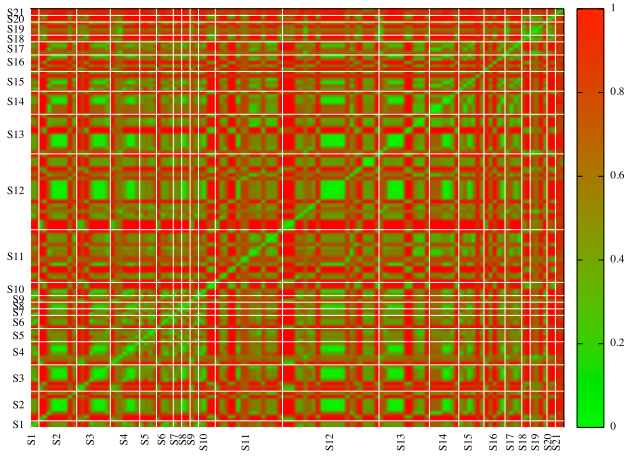
 ISBN: 978-1-61208-670-5



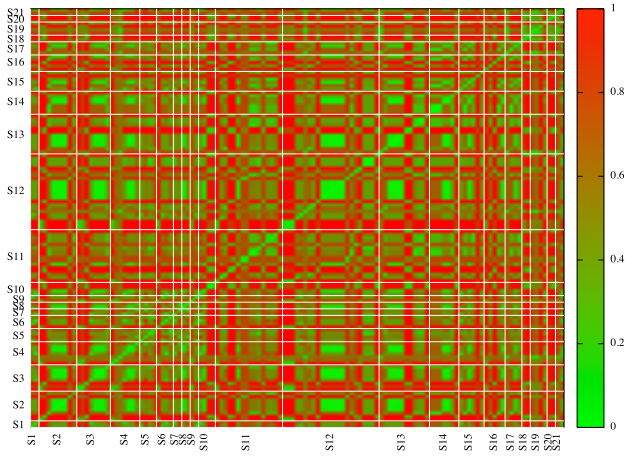
a) Hausdorff with Euclidean distance



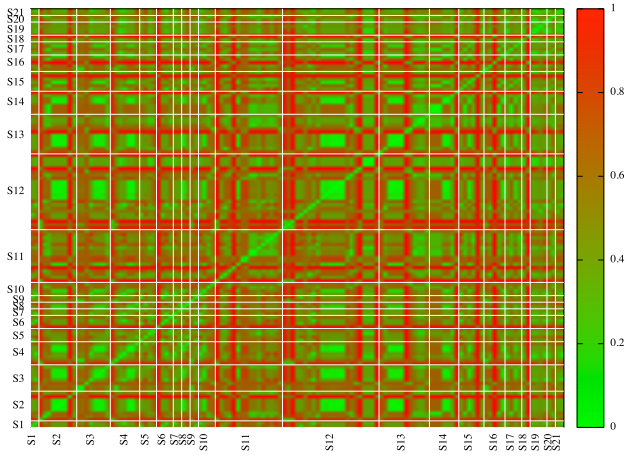
b) Hausdorff with Euclidean distance and normalised data



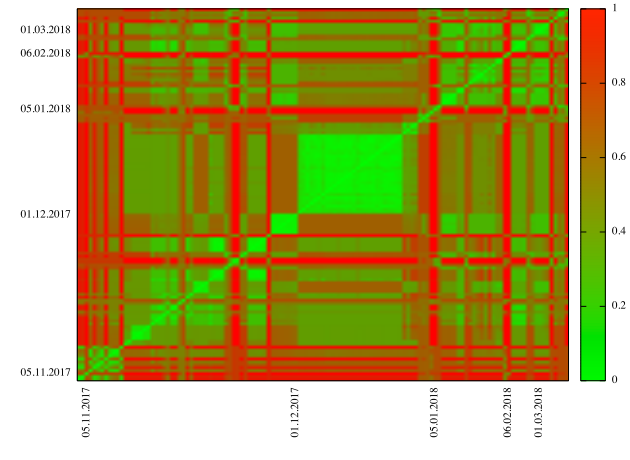
c) Hausdorff with Chebychev distance and normalised data



d) Hausdorff/Chebychev, normalised and filtered columns



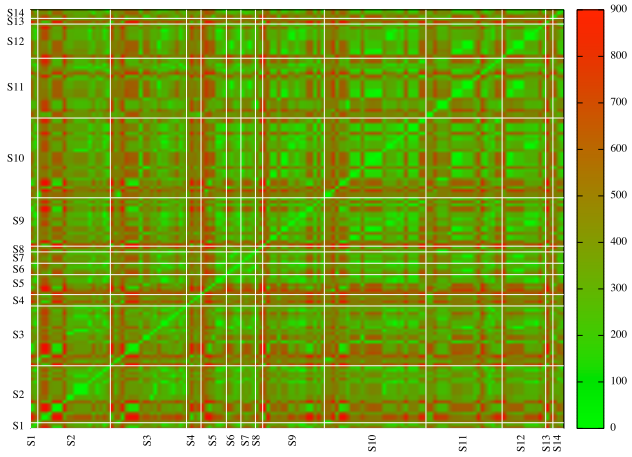
e) Hausdorff / Chebychev, normalised and filterd columns and longer fingerprint



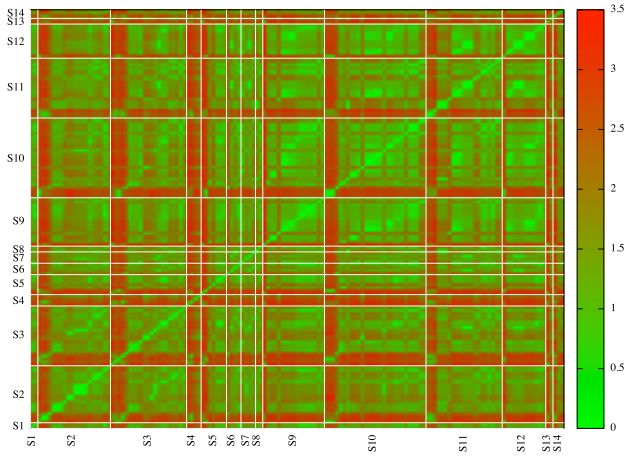
f) Same settings as e) but ordered by time

Figure 6. Results for Dataset 5

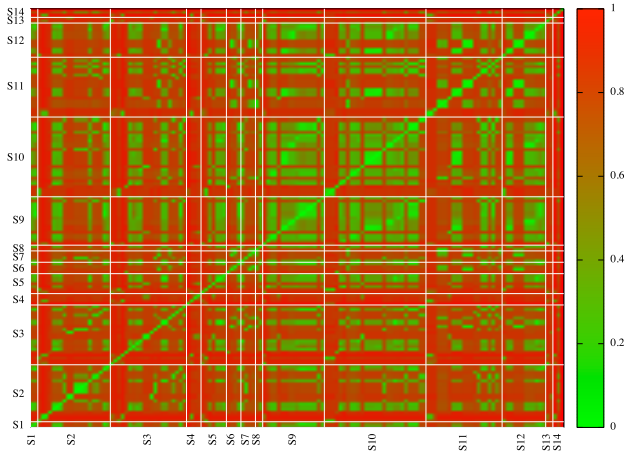
 ISBN: 978-1-61208-670-5



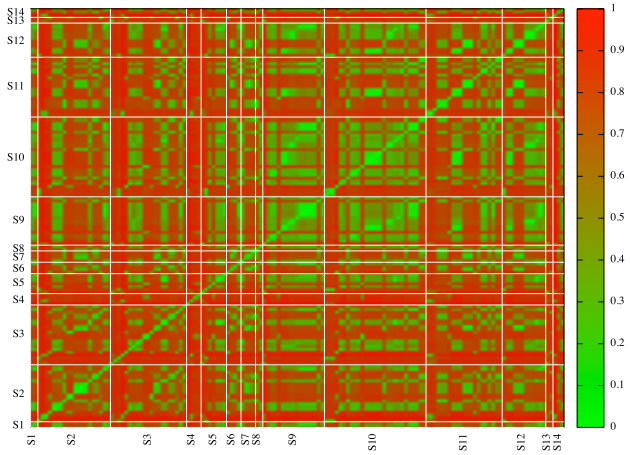
a) Hausdorff with Euclidean distance



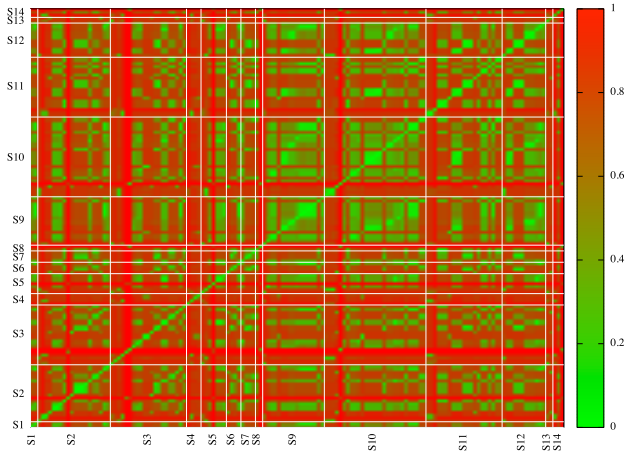
b) Hausdorff with Euclidean distance and normalised data



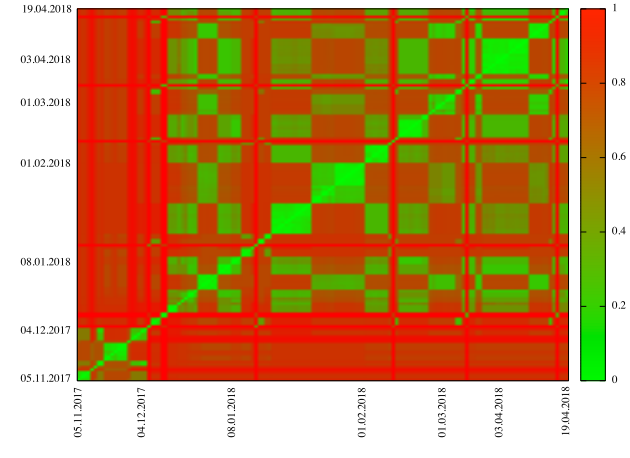
c) Hausdorff with Chebychev distance and normalised data



d) Hausdorff/Chebychev, normalised and filtered columns



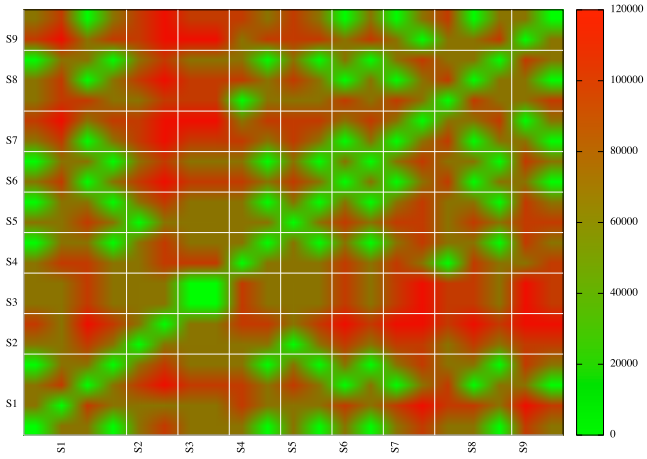
e) Hausdorff / Chebychev, normalised and filterd columns and longer fingerprint



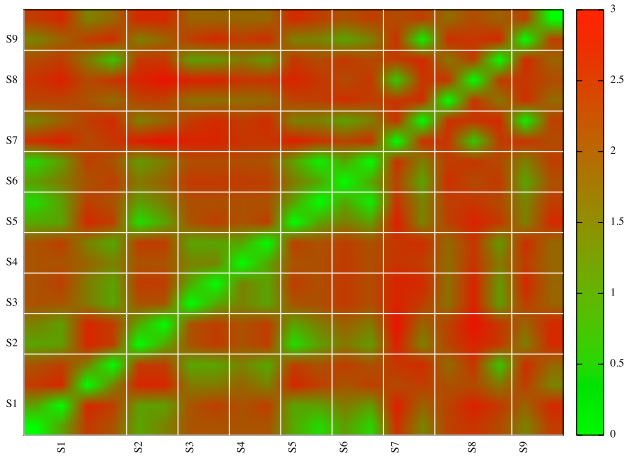
f) Same settings as e) but ordered by time

Figure 7. Results for Dataset 6

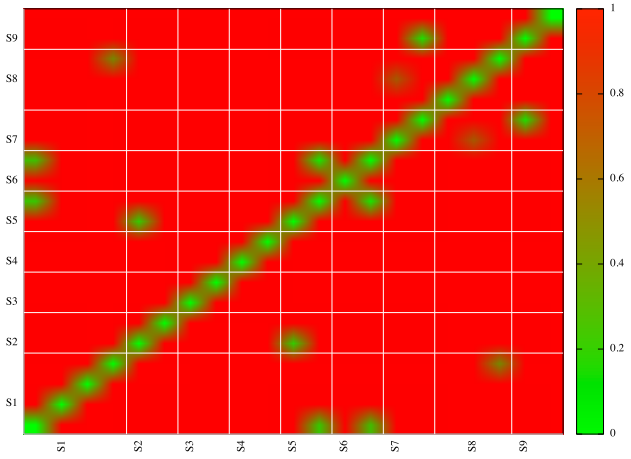
 ISBN: 978-1-61208-670-5



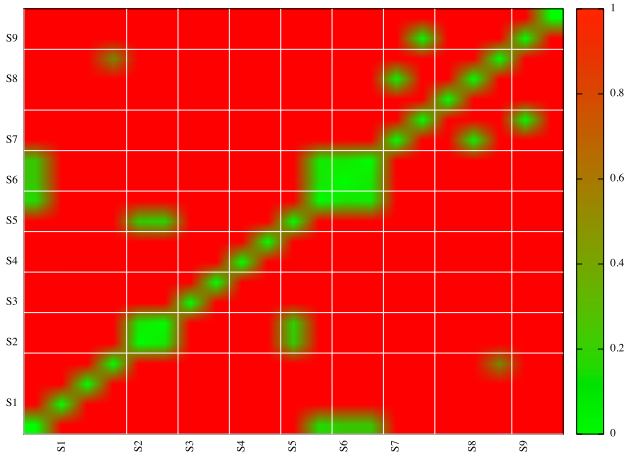
a) Hausdorff with Euclidean distance



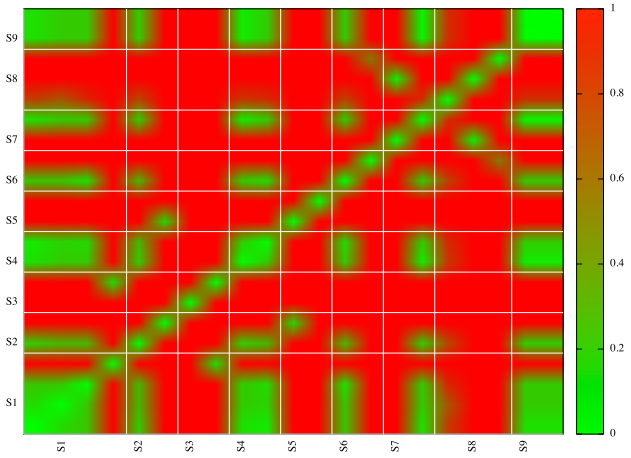
b) Hausdorff with Euclidean distance and normalised data



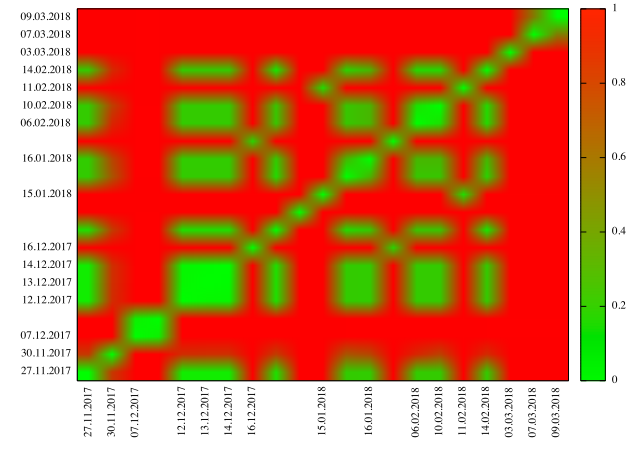
c) Hausdorff with Chebychev distance and normalised data



d) Hausdorff/Chebychev, normalised and filtered columns



e) Hausdorff / Chebychev, normalised and filterd columns and longer fingerprint



f) Same settings as e) but ordered by time

Figure 8. Results for Dataset 7