Abstracts of the FLLL/SCCH Master and PhD Seminar

Room 010, Software Park Hagenberg

May 7, 2004

Software Competence Center Hagenberg
Hauptstrasse 99
A-4232 Hagenberg
Tel. +43 7236 3343 800
Fax +43 7236 3343 888
www.scch.at

Fuzzy Logic Laboratorium Linz-Hagenberg
Hauptstrasse 99
A-4232 Hagenberg
Tel. +43 7236 3343 431
Fax +43 7236 3343 434
www.flll.jku.at

Program

Session 1 (Chair: Susanne Saminger) 9:00–10:30

9:00 Mario Drobics:
Feature selection using bacterial optimization

9:30 Thomas Biringer:
Knowledge based methods in management

10:00 Werner Groißböck:
System identification based on variable selection and genetic programming III

10:30 Coffee break

Session 2 (Chair: Ulrich Bodenhofer) 11:00–12:30

11:00 Daniel Soukup:
Print layer identification in series of multi layer print images

11:30 Leila Muresan:
Incremental version of the IPAN method

12:00 Michaela Kröppl, Alwin Pichler:
Presentation of the high-tech incubator program `tech2b`

s c c h
software competence center
hagenberg

Technical Report
SCCH–TR–0408

Feature Selection using Bacterial Optimization

J. Botzheim
Department of Telecommunication and Media Informatics,
Budapest University of Technology and Economics,
botzheim@alpha.tmit.bme.hu

M. Drobics
Software Competence Center Hagenberg,
A-4232 Hagenberg, Austria
mario.drobics@scch.at

L. T. Kóczy
Department of Telecommunication and Media Informatics,
Budapest University of Technology and Economics,
koczy@tmit.bme.hu

Abstract — When creating regression models from data the problem arises that the complexity of the models rapidly increases with the number of features involved. Especially in real world application where a large number of potential features are available, feature selection becomes a crucial task. In this paper we will present a novel approach to feature selection which uses bacterial optimization to identify the optimal set of features with respect to a given learning problem *and* a given learning algorithm. This approach ensures high accuracy and significantly increases interpretability of the resulting models.

Key words — *feature selection, bacterial optimization*

Proc. Int. Conf. on Information Processing
and Management of Uncertainty in
Knowledge-Based Systems 2004, to appear

Kplus
Kompetenzzentren-Programm

1 Introduction

To create regression models from data several methods like statistical regression, neural networks, or regression tree methods [10] exist. The problem, however, arises that the complexity of these models rapidly increases with the number of features involved. This causes two major problems: On the one hand, some features may have a very low bias but a high variance and mislead the regression methods. On the other hand a large number of predictors decreases interpretability, as the major influences are likely to be shadowed by other unimportant features. Furthermore, taking measurements is often a time consuming and costly task. Reducing the number of measurements (features) used is therefore an important design goal.

Dimension reduction or feature selection can be used to reduce the dimensionality of the original state space (i.e. to reduce the number of features under investigation). While dimension reduction methods like Principal Component Analysis (PCA) [2] use projection methods which often cumber interpretation of the resulting models, feature selection methods aim to identify the most relevant features out of the original set of features [5, 8].

Nature inspired some evolutionary optimization algorithms suitable for global optimization of even non-linear, high-dimensional, multi-modal, and discontinuous problems. The original genetic algorithm was developed by Holland [7] and was based on the process of evolution of biological organisms. It has been successfully applied to the problem of feature extraction [11, 12]. The drawback of genetic algorithms, however, is that they require a large number of iterations. A more recent approach is the bacterial evolutionary algorithm. This gives an alternative to other algorithms because it is simpler and it is possible to reach lower error levels within a short time. This method includes two new operations inspired by the microbial evolution phenomenon. The bacterial mutation operation which optimizes the chromosome of one bacterium, and the gene transfer operation which transfers information between different bacteria within the population.

In this paper we will present a novel approach to feature selection using bacterial optimization. First, the general problem of feature selection is described in Section 2. Then, in Section 3 the bacterial algorithm is introduced and we show how this method can be applied to the problem of feature selection. In Section 4 some simulation results are presented to show the potential of this new approach.

2 Feature selection

Let us consider the following general setting: Let $\mathcal{U} = X \times Y$ be our universe of discourse, where X is the n-dimensional input space and Y the one dimensional output space. The overall goal of the learning process is to find a function $f : X \mapsto Y$ which models the inherent relation between the input and the output space. In machine learning this function f is induced from a set of training samples $\mathcal{S} \subset X \times Y$ by minimizing an error measure $E(f, \mathcal{S})$. Usually the average squared error is used

$$E(f, \mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{x},y)\in\mathcal{S}} \big(f(\mathbf{x}) - y\big)^2.$$

An increase in the dimensionality of the input space increases the complexity of this learning problem. When features with only minor or no relation at all to the output space are involved,

the resulting function f might tend to overfit the training data. Although various methods exist to overcome these problems, it is often more efficient to reduce the number of features in advance.

Feature selection can be described as the task of identifying an optimal subset of m out of the available n features. The resulting subset of m features is then used to compute a function $\bar{f}$, which maps from the m-dimensional input space $\bar{X}$ to the output space Y.

In the following we will discuss a so-called *wrapper* method for feature selection which utilizes the regression learner as a black box to score a subset of variables according to their overall predictive power. In contrast to *filter* methods which aim to identify the most relevant features before the actual learning step, wrapper methods are able to identify relevant combinations of input features, too. As the problem of finding an optimal subset of features is known to be NP-hard [1], intelligent search strategies are essential. Such intelligent strategies provide suboptimal solutions, however, they approximate the optimum rather well.

3 Bacterial optimization

There are several optimization algorithms which were inspired by the processes of evolution. These processes can be easily applied in optimization problems where one individual corresponds to one solution of the problem. An individual can be represented by a sequence of numbers that can be bits as well. This sequence is called *chromosome*, which is nothing else than the individual itself. In bacterial algorithms, the bacteria can transfer genes to other bacteria. This mechanism is used both in the *bacterial mutation* and the *gene transfer* operations. The latter substitutes the genetic algorithms crossover operation, so information can be transferred between different individuals.

3.1 The encoding method

In bacterial algorithms, one bacterium $\xi_i, i \in I$ corresponds to one solution of a given problem. First, we have to define how a solution is encoded in such a bacterium (chromosome). For the task of selecting m features from a set of n features ($m \leq n$), the bacterium consists of a vector of integers $\xi_i = \{\xi_i^1, \ldots, \xi_i^m\}, 1 \leq \xi_i^k \leq n$, where $\xi_i^k \neq \xi_i^l$ for $k \neq l$.

3.2 The evaluation function

Similar to genetic algorithms the fitness of a bacterium ξ_i is evaluated using an *evaluation function* $\phi(\xi_i)$. The choice of this evaluation function is problem dependent. For the task of feature selection we use the features encoded in bacterium ξ_i and the training data set $\mathcal{S}$ to compute a regression model f_i according to:

$$f_i(\mathbf{x}) : X_{\xi_i^1} \times \ldots \times X_{\xi_i^m} \mapsto Y.$$

The evaluation function is then computed as the average squared error of the input-recall behavior of this model on test data set $\mathcal{T} \subset X \times Y$:

$$\phi(\xi_i) = E(f_i, \mathcal{T}).$$

3.3 The evolutionary process

The basic algorithm consists of three steps [3, 9]. First, an initial population has to be created randomly. Then, bacterial mutation and gene transfer are applied, until a stopping criteria is fulfilled. The evolution cycle is summarized below:

Bacterial Optimization

create initial population
do {
  apply bacterial mutation
  apply gene transfer
} **while** stopping condition not fulfilled
return best bacterium

3.4 Generating the initial population

First an initial bacterium population of N_{ind} bacteria $\{\xi_i, i \in I\}$ is created randomly ($I = \{1, \ldots, N_{\text{ind}}\}$). Figure 1 shows a bacterium ξ_i with $n = 50$ and $m = 5$.

44	17	36	2	7
ξ_i^1	ξ_i^2	ξ_i^3	ξ_i^4	ξ_i^5

Figure 1: A single bacterium

3.5 Bacterial mutation

Bacterial mutation is applied to all bacteria $\xi_i, i \in I$. First, N_{clones} copies (clones) of the bacterium are created.

$$\xi_{i,j} = \xi_i, \qquad \forall j : 1 \leq j \leq N_{\text{clones}}$$

Then, in each clone $\xi_{i,j}$ a random part k of the chromosome is replaced by a random number smaller or equal than n ($\xi_{i,j}^k = \text{Random}[n]$). When we change a part of a bacterium, we must take care that the new part is unique within the selected bacterium $\xi_i^k \neq \xi_i^l$ for $k \neq l$. Next, all the clones and the original bacterium are evaluated using the evaluation function $\phi(\xi)$. The bacterium with the best evaluation result is used to transfer the mutated part to the other individuals. This cycle is repeated for the remaining parts, until all parts of the chromosome have been mutated and tested. At the end, the best bacterium is kept and the remaining N_{clones} are discharged. Figure 2 shows an example mutation for $N_{\text{clones}} = 3$.

3.6 Gene transfer

The bacterial mutation operator optimizes the bacteria in the population. Often, however, this is not enough as we need to provide a possibility for some information flow within the population.

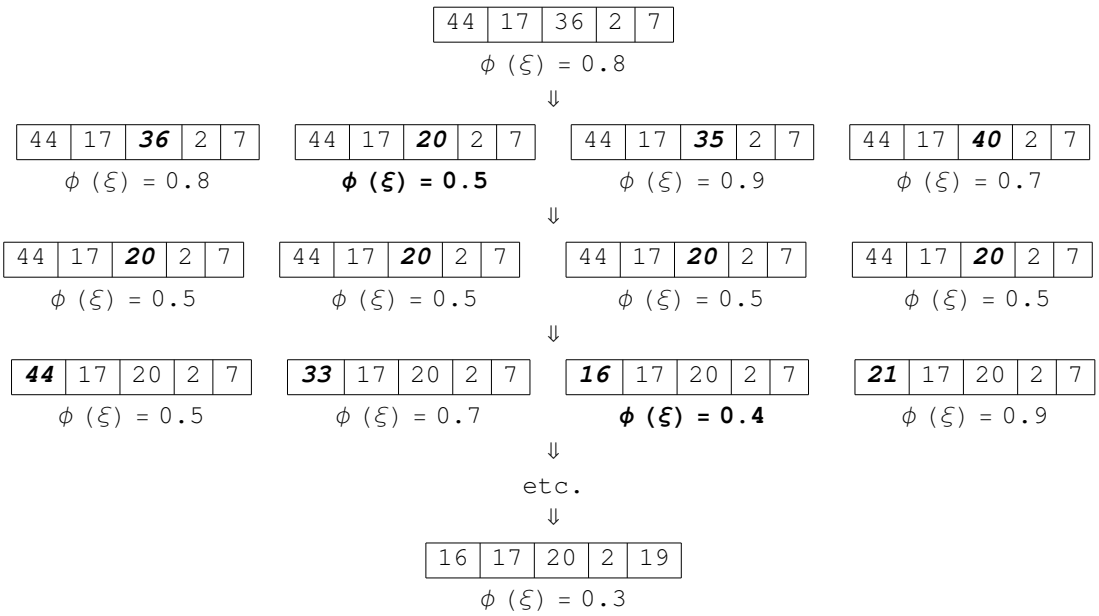


Figure 2: Bacterial mutation

Using the gene transfer operator, the recombination of genetic information between two bacteria is possible.

1. First, the population must be sorted and divided into two halves according to their evaluation results. The bacteria with a higher evaluation are called superior half, the bacteria with a lower evaluation inferior half.

2. Then one bacterium is randomly chosen from the superior half and another from the inferior half. These two bacteria are called the source bacterium, and the destination bacterium, respectively.

3. A part from the source bacterium is randomly chosen and this part is used to overwrite a random part of the destination bacterium if the source part is not already in the destination bacterium.

Gene transfer is repeated N_{inf} times, where N_{inf} is the number of "infections" per generation. As a default value N_{inf} is set to $N_{\text{ind}} - 1$. Figure 3 shows an example for the gene transfer operations ($N_{\text{ind}} = 4, N_{\text{inf}} = 3$).

3.7 Stopping condition

If all individuals in the population are equal or the maximum number of generations N_{gen} is reached then the algorithm ends, otherwise it returns to the bacterial mutation step. Typically, a small number of generations (below 20) already leads to good results.

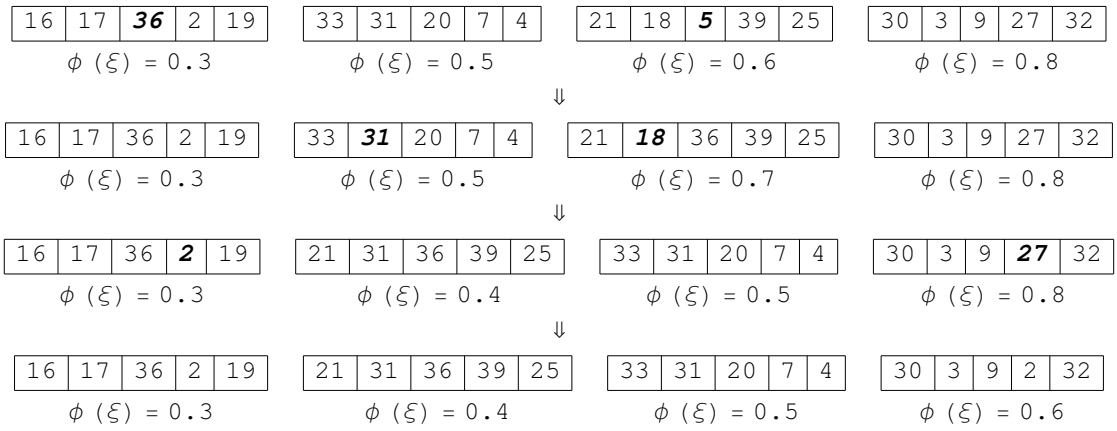


Figure 3: Gene transfer

4 Simulation results

The bacterial optimization used for the following simulations was realized in Mathematica. We used three different regression methods to analyze the behavior of the algorithm—a simple least-squares optimization and two fuzzy rule base induction systems. We applied the bacterial optimization for all of the regression methods to find a solution for three high-dimensional problems.

The first test function was defined over a 20-dimensional data space $[0,1]^{20}$ according to:

$$f_{20}(\mathbf{x}) = x_1 x_2^2 x_{13}^3 - x_{20} + 5\sin(x_{16}) - 25\cos(x_5 x_{18}) + \exp(x_3 x_5) + x_4 x_{19} + x_{10}^2 + x_{11}^5.$$

The second test function was defined over $[0,5]^{50}$ according to:

$$f_{50a}(\mathbf{x}) = x_1 x_{40}^2 + 0.01\, x_{12} + 0.01\, x_{15} + 3.1 x_{49}.$$

The third test function was defined over $[1,2]^{50}$ according to:

$$f_{50b}(\mathbf{x}) = x_1 + x_2^{\frac{1}{2}} + x_3 x_4 + 2\exp(2(x_5 - x_6)) + x_7 x_8 x_9 - \frac{x_{10}}{x_{11} x_{12}} + 0.5\, x_{13}^3 + x_{14}^2 x_{15} - x_{16} - x_{17} + 50\,\frac{x_{18} + x_{19}}{x_{20}} + x_{46}$$

We created 500 training samples for each test function by assigning each input dimension a random number with equal distribution. Each of the chosen functions have random behavior in the remaining dimensions generated by a random generator. Many simulations were made (see Table 1), some typical results are discussed in this section.

4.1 Statistical fit function

In the first example we created a prediction model by computing a least-squares fit to the data as a linear combination of the input features.

Although good results have been achieved with the initial settings for the first test function, a better solution was found when increasing the number of clones in the second run. In the third and fourth run, the number of goal dimensions was increased to ten, which significantly improved the obtained results. When using two bacteria, the solution was found earlier than with only one bacterium. In the first case all individuals were the same in the third generation.

For the second test function, we tried to identify the three most important dimensions. In the first run, we used one bacterium and five clones, in the second four bacteria, four clones and two gene transfer operations in each generation. The optimal solution was found in the fifth and fourth generation, respectively. The solution found was $\{40, 49, 1\}$ and from the definition of the function it is easy to see, that these are in fact the most important variables. When the bacterium length was set to five two additional variables were found, but the overall performance only slightly increased. The learning curves for this problem are shown in Figure 4.

With the initial setting used for the third test function, 16 generations were needed. When the number of clones was increased, the optimal solution was found already in the eighth generation. By increasing the number of individuals and by applying gene transfer, the algorithm was able to find the optimal solution in six generations. In the second case we set the length of one bacterium to ten. When using four clones the solution was not optimal. A lower error level was reached using six clones. When using four bacteria instead of one, the same solution was found already in the fifth generation, and all bacteria were equal in the 20th generation.

4.2 RENO optimization

In this example we created a prediction model using a fuzzy rule induction method called *RENO* [6]. RENO first computes a set of equally distributed fuzzy sets for each input dimension. Then for each element of the cartesian product of these fuzzy sets a TSK rule is created. Finally, the resulting rule base is optimized using a regularized numeric optimization technique to tune the fuzzy sets and the linear approximation on the right-hand side. Although this methods leads to very accurate and stable models, it is limited to low dimensional problems ($n \leq 8$) as the number of rules increases exponentially with the number of dimensions involved. Therefore feature selection is crucial when applying it to higher dimensional problems.

When applying RENO optimization, we always tried to identify the three most important variables. With the first test function, the same solutions were found in all runs. In the case where we applied not only one individual, all individuals became identical after four generations. The best individual found was $\{18, 5, 16\}$ which are indeed the most important variables of this function.

For the second test function, no satisfying solution was found in a first attempt. After increasing the number of clones to four, however, a good solution was found. Increasing the number of bacteria and applying gene transfer led to the same solution. The learning curves for this problem can be found in Figure 5.

For the third test function an optimal solution was found using four clones. When using four individuals, the solution was found earlier. Increasing the number of clones, however, misled the algorithm and no good solution was found.

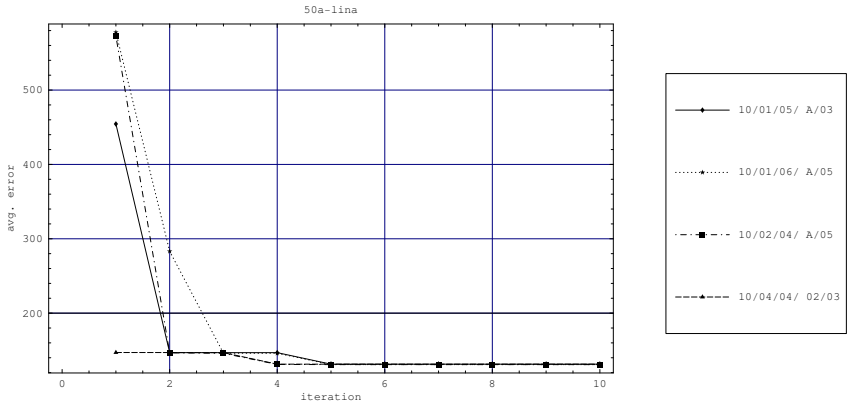


Figure 4: Simulation result for the second function using linear approximation

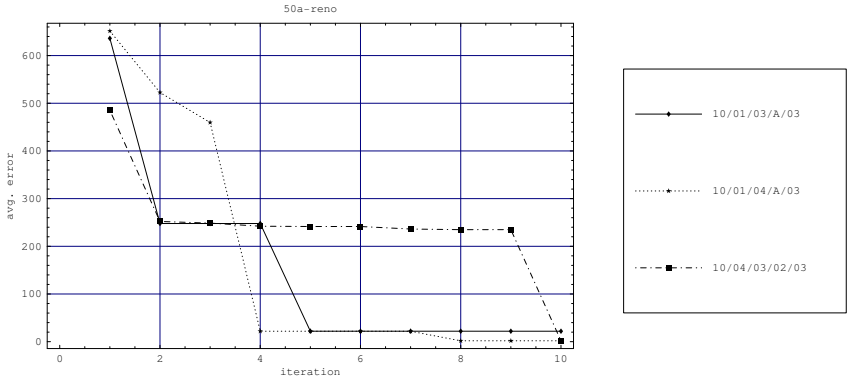


Figure 5: Simulation result for the second function using RENO

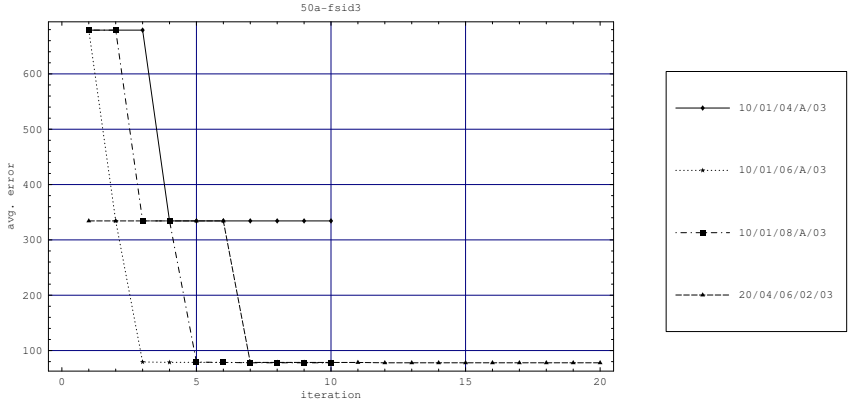


Figure 6: Simulation result for the second function using FS-ID3

Table 1: Settings of the parameters

f	method	N_{gen}	N_{ind}	N_{clones}	N_{inf}	m	error	b. gen.	best bacterium
f_{20}	LINA	10	1	4	0	5	1.41	3	{18, 16, 20, 5, 10}
f_{20}	LINA	10	1	6	0	5	1.40	3	{18, 10, 5, 16, 11}
f_{20}	LINA	10	1	4	0	10	1.25	8	{4, 5, 16, 20, 3, 18, 11, 10, 17, 15}
f_{20}	LINA	10	2	4	1	10	1.25	1	{4, 18, 5, 20, 3, 15, 17, 16, 10, 11}
f_{20}	RENO	10	1	4	0	3	0.36	3	{5, 18, 16}
f_{20}	RENO	20	1	6	0	3	0.36	1	{5, 16, 18}
f_{20}	RENO	20	2	6	1	3	0.36	3	{18, 5, 16}
f_{20}	FS-ID3	10	1	4	0	5	1.99	3	{13, 14, 16, 5, 18}
f_{20}	FS-ID3	10	1	6	0	5	1.99	3	{16, 13, 5, 18, 9}
f_{20}	FS-ID3	10	1	4	0	10	2	2	{20, 6, 16, 5, 15, 10, 14, 18, 13, 3}
f_{20}	FS-ID3	10	2	4	1	10	2.06	1	{8, 18, 12, 16, 6, 5, 4, 3, 7, 10}
f_{50a}	LINA	10	1	5	0	3	131.38	5	{49,1,40}
f_{50a}	LINA	10	4	4	2	3	131.38	4	{40,49,1}
f_{50a}	LINA	10	1	6	0	5	130.71	10	{9,4,40,49,1}
f_{50a}	LINA	10	2	4	1	5	130.78	6	{40,9,27,1,49}
f_{50a}	RENO	10	1	3	0	3	21.93	9	{1,8,40}
f_{50a}	RENO	10	1	4	0	3	1.82	8	{1,49,40}
f_{50a}	RENO	10	4	3	2	3	1.82	10	{49,40,1}
f_{50a}	FS-ID3	10	1	4	0	3	334.29	4	{37,33,40}
f_{50a}	FS-ID3	10	1	6	0	3	77.87	10	{40,24,1}
f_{50a}	FS-ID3	10	1	8	0	3	77.87	7	{40,24,1}
f_{50a}	FS-ID3	20	4	6	2	3	77.87	12	{24,40,1}
f_{50b}	LINA	20	1	4	0	5	29.49	16	{19, 18, 20, 14, 6}
f_{50b}	LINA	20	1	6	0	5	29.49	8	{19, 14, 20, 18, 6}
f_{50b}	LINA	20	4	6	2	5	29.49	6	{20, 6, 18, 14, 19}
f_{50b}	LINA	20	1	4	0	10	25.22	11	{14, 17, 5, 6, 9, 20, 8, 15, 19, 18}
f_{50b}	LINA	20	1	6	0	10	24.58	14	{19, 6, 8, 15, 14, 5, 18, 9, 13, 20}
f_{50b}	LINA	20	4	6	2	10	24.58	5	{15, 18, 20, 5, 9, 14, 19, 13, 6, 8}
f_{50b}	RENO	10	1	4	0	3	10.15	10	{19, 18, 20}
f_{50b}	RENO	10	1	6	0	3	97.39	10	{37, 19, 20}
f_{50b}	RENO	10	4	4	2	3	10.15	8	{20, 18, 19}
f_{50b}	FS-ID3	20	1	4	0	5	57.51	20	{20, 19, 39, 18, 25}
f_{50b}	FS-ID3	10	4	4	2	5	55.71	8	{19, 18, 20, 36, 35}
f_{50b}	FS-ID3	20	1	4	0	10	54.4	16	{40, 26, 19, 17, 30, 25, 18, 20, 9, 46}

4.3 Decision tree optimization

Finally, we used the proposed method to optimize the performance of an inductive fuzzy decision tree learning method *FS-ID3* [4]. This method uses a top-down approach to create a decision tree, which can be applied for classification as well as for regression problems. Although this method is capable of dealing with a large number of input features, stability and interpretability can be improved using only a subset of the available features.

For the first test function the solution was found quickly both with length five and ten. For the second function, more number of clones led to a better solution. For the third function, the trials where more individual were used gave better result. These examples illustrate the importance of the number of generations, as the optimal solution may only be reached in a latter generation.

4.4 Conclusion

The examples given illustrate that the proposed method can be used in combination with various existing methods for regression as well as for classification tasks.

Compared to classical genetic algorithm, the bacterial evolution process converges faster. Another advantage of the method is that the gene transfer operator can be realized easier than the crossover operator in GA, because in the crossover operation the multiple appearing of all the elements in the whole chromosome has to be checked, while in the crossover operation only the transferred part needs to be checked.

In our simulations it turned out, that increasing the number of generations and the number of individuals also increases the performance of the algorithm. Increasing these parameters, however, causes additional computational effort.

Finding the optimal number of clones is a more subtle problem. A small number of clones (below four) causes the algorithm to get stuck in local minima as no new information is available from other clones. If the number of clones is too large, the algorithm converges too fast and might get stuck in a local minima, too.

The choice of the number of gene transfers is correlated with the number of individuals. Using N individuals without gene transfer is similar to running the algorithm with only one individual but N times. The gene transfer operator enables interaction between the bacteria in the population. In some way, the local solutions are compared and enable the algorithm to find the global optimum.

5 Outlook

Future work will be concerned with extending the bacterial optimization to identify not only the optimal subset of a given size, but to find the optimal size of this subset, too. Additionally, we intend to investigate different bacterial operators which change not only one number, but a longer part of the chromosome to avoid local optima. Furthermore, we plan to implement a sub-sampling strategy to reduce the computational complexity of the evaluation function. Finally, we want to investigate more effective stopping criteria.

Acknowledgements

This research was supported by the National Scientific Research Fund OTKA T034233, T034212 and T043177, a Széchenyi University Research Grant 2004, the National Research and Development Project Grant NKFP-2/0015/2002 and CEEPUS SK-42 Grant.

This work has partly been done in the framework of the K*plus* Competence Center Program which is funded by the Austrian Government, the Province of Upper Austria, and the Chamber of Commerce of Upper Austria.

References

[1] E. Amaldi and V. Kann. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209(1–2):237–260, 1998.

[2] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[3] J. Botzheim, B. Hámori, L. T. Kóczy, and A. E. Ruano. Bacterial algorithm applied for fuzzy rule extraction. In *Proc. Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 1021–1026, Annecy, France, 2002.

[4] M. Drobics and U. Bodenhofer. Fuzzy modeling with decision trees. In *Proc. 2002 IEEE Int. Conf. on Systems, Man and Cybernetics*, Hammamet, Tunisia, October 2002.

[5] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *JMLR*, 4:1157–1182, 2003.

[6] J. Haslinger, U. Bodenhofer, and M. Burger. Data-driven construction of Sugeno controllers: Analytical aspects and new numerical methods. In *Proc. Joint 9th IFSA World Congress and 20th NAFIPS Int. Conf.*, pages 239–244, Vancouver, July 2001.

[7] J. H. Holland. *Adaptation in Natural and Artificial Systems*. The MIT Press, Cambridge, MA, first MIT Press edition, 1992. First edition: University of Michigan Press, 1975.

[8] M. Last, A. Kandel, and O. Maimon. Information-theoretic algorithm for feature selection. *Pattern Recognition Letters*, 22:799–811, 2001.

[9] N. E. Nawa and T. Furuhashi. Fuzzy system parameters discovery by bacterial evolutionary algorithm. *IEEE Tr. Fuzzy Systems*, 7:608–616, 1999.

[10] J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[11] H. Vafaie and K. De Jong. Genetic algorithms as a tool for feature selection in machine learning. In *Proc. 12th IEEE Int. Conf. on Tools with Artificial Intelligence*, pages 200–204, Arlington, VA, 1992.

[12] D. Whitley, J. R. Beveridge, C. Guerra-Salcedo, and C. Graves. Messy genetic algorithms for subset feature selection. In Thomas Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA97)*, San Francisco, CA, 1997. Morgan Kaufmann.

Knowledge Based Methods in Management

Thomas Biringer

E-Mail : Thomas.Biringer@magnasteyr.com

Abstract: --- In the car production industry the time to market is in the future a key indicator for the ability to solve a lot of money in reducing the time to market. In this case it is necessary to use new methods to solve the obtained knowledge from running projects in a knowledge database for further projects. In this case I will show a possible way to secure obtained knowledge with using cybernetic model and machine learning.

Key words -- machine learning, cybernetic

1 Introduction

Reffering to the topic „start-up-management“ there are no essays which ones demonstrate the whole process incipient with development up to start of production, inclucing all involved divisions. There are no fruitful disquisitions beyond the extensive cohesions among development, logistics, serialproduction und their interact. So there is no basis for controlling a startup and for rating the existing risk to meet the deadline, achieve quality and expenses. Presently there is no conclusive knowledge base for a start of production. It´s interesting to regard the start up concerning „novel product, new craft, new employees“ from the point of view of the producing plant. To my mind it´s witted – on the basis of the experience of the successful start up – to create a knowledge base for future startups and modeling the start up with new techniques for knowledge engineering and to compare the results of modeling to the values of the real start up.

Within the start up it became clear, that - due to the huge complexness of the organisational structure - the present „Technomorphe Managementsystem“ is on upper bound respectively the networked processes can´t be acquired conclusive and the parameters as a whole don´t flow into decision making. Due to the nonlinear mutual processinteractions and the non-controllable complexity there often arises a limited consideration for problems, whereby the management is pushed into a reactive part respectively in worst case, it elects for a non-target-oriented measure.
Due to the ever-shortend development-period it comes as well to a intensive integration of processes. Thus it´s essential to find the primal factors of success to control the start of production.
To find the essential factors (canals with huge information content), there should be applied new treatments from different fields of knowledge (such as Maschinenlernen) and be checked on their capability.
In „Technomorphen Systemen“, where it´s assumed, that enough information is available, it´s applied at best in clear systems, so it´s needful to arrange a “Systemisches Modell”, what it used as base for controllability “Maschinenlernen”.
In line with „Maschinenlernen“ it´s essential to use a algorithm, what isn´t just adopt to anticipate information, but to bring out a linguistic context from observed facts out of the basic start-up. By this means it gets possible to operators, to discover abnormal cohesions out of the basic start-up and if necessary taking corrective action.

2 Behaviours of complex systems

2.1 Constructivist Technomorphe System

Optimality

The constructivist approach is the ambition on optimality, therefore complete information is essential. [1]
From the point of the “technomorphen“ view, optimisation is equal to elimination of flexibility (in according to engine: An engine can be construct and optimized the better the more firm its input is and the lighter fluctuation due to quality and quantity. q.v.: Out of ideal batch sizes in production result therefore essentially low limits of variation and thus the loss of response-flexibility). **There never exists enough knowledge for decissions.**

Profit-maximizing

For this type „advantage-maximizing-cerebration“ is immanent. It´s not about the discussion on details but about the question, what important factors admit to minimize the danger of systemic wrong-decissions. [2]
To realize profit it´s necessary to have profit-potential as well as future profit-potential in condition to earn future-protits.

Sufficient information-base

In line with the construktivist typ of theory it´s normally assumed, that the information-base is in the main sufficient for the solution of discussed problems. The blank spaces of information are filled with subjektive probability-estimates. This pretended gained security applies in context just in bounded real systems. [3]

2.2 System- evolutionary systems

Controlability

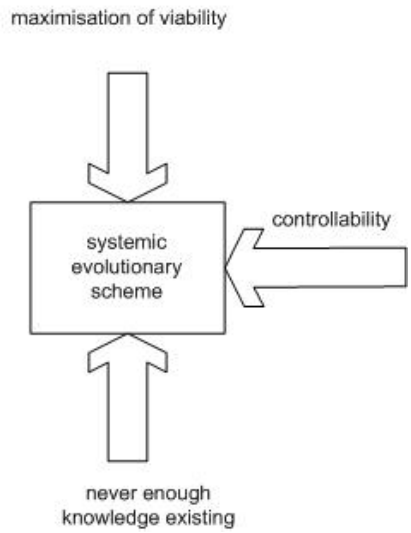
Cosindering the premise that a permanent adaption to situations is necessary, optimization can only make sense on a meta level. Therefore not a single state of adaption has to be optimised but if anything at all the adaptiveness has to be optimised. The contolability and the manageability of a company has to be optimised . [3]
Pay attention: Ideal organizations from the past may be crucial obstacles for modern adaptions.

Ability to live maximization

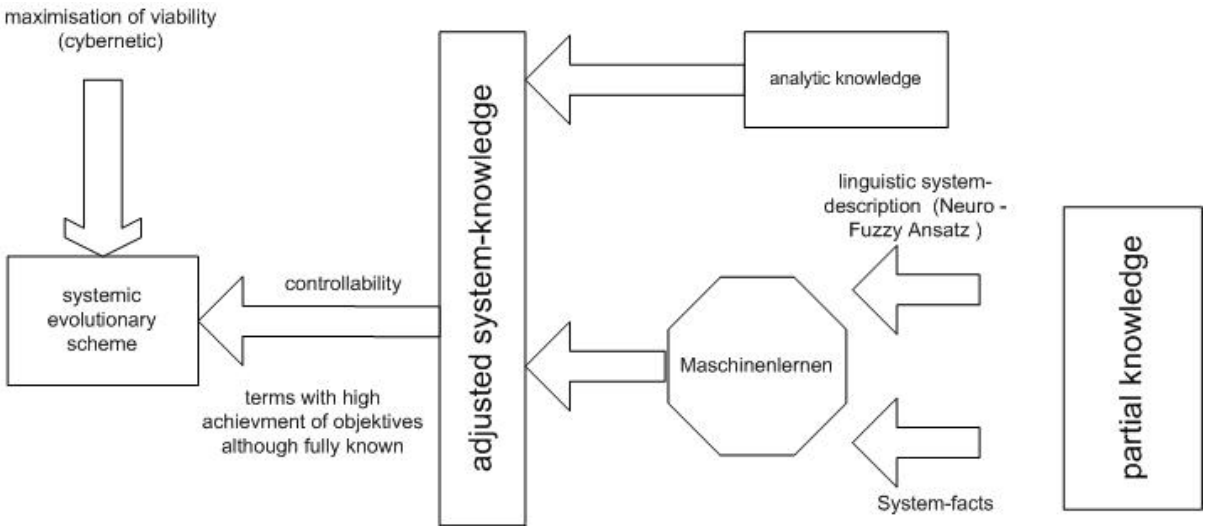
The system- evolutionary approach aims at companies in particular and systems in general. The ability to live of companies and systems is not meant to be a formal criteria but to be considered in the content of an empiric problem. There are problems that can be solved and others that can't. Therefore the ability to live can be seen in relation to the controllability of a company, since the ability to live is a preliminary proof, that the complexity of the system can be controlled . [4]

Never enough knowledge existing

The systemic type explicitly assumes, that there is not enough information for legitimating decisions. → The association to legitimate a decision is wrong. [3] This teaches us to possibly make decisions, of which most of the consequences can be withdrawn.



3 Use of machine learning methods



Approach of machine learning without linguistic description

$$T = \{(x_1, y_1), (x_2, y_2), ...(x_L, y_L)\}$$

Sought after: Function f with

$$\hat{f}(x) = \hat{y} \approx y$$

Target: Good prediction

$$G = \{(x_1, y_1), (x_2, y_2), ...(x_M, y_M)\}$$

Because of the enhancement of machine learning about the linguistic description the system knowledge is expanded from pure data analyze with interpretable knowledge.

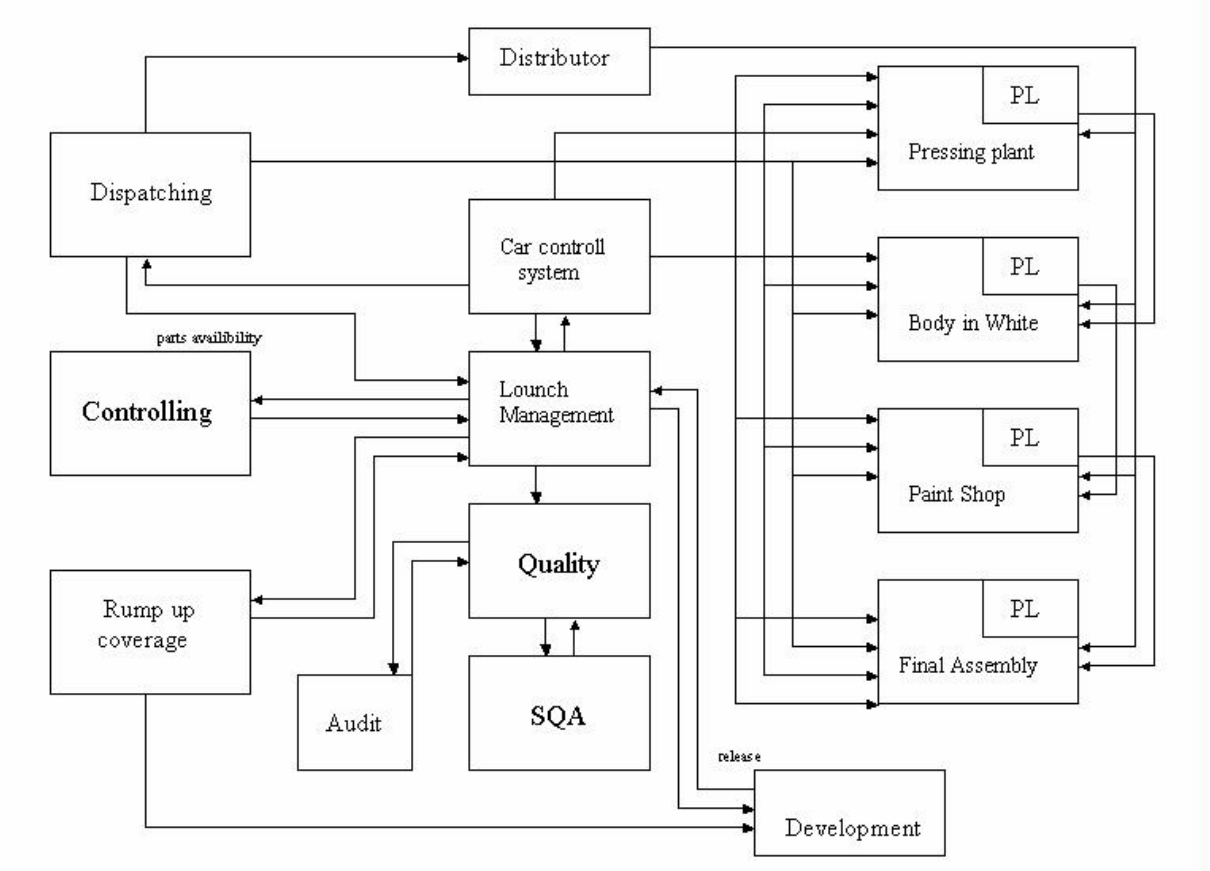
$$\hat{f}(x) = \hat{y} \approx y \quad \text{with} \quad \hat{y} \approx y + \text{interpretation } y$$

Thereby you are able to calculate the inversion of this problem. Therefore it is possible to modify the input data because of the knowledge of the required initial state.

$$\hat{f}^{-1}(\hat{y}) = x \quad \text{with} \quad \hat{y} \approx y + \text{interpretation } y$$

For this purpose it is necessary to describe the impact data and parameters from the system with exact linguistic data.

Complex lunch structure
Description of all input data in linguistic way. For handling this problem an approach of a hierarchical controller design will be used.



4 Conclusion

This paper is written at the beginning of my PhD-thesis and shows a possible way to find dates for a knowledge based model.

- an approximation with accuracy of forecast based on existing cognition as knowledge base for decision making.
- a systemic evolutionary model in consideration of cybernetic aspects in adoption of learning structures as knowledge base, what is available for future start-ups.

5 References

[1] Fredmund Malik *Systemische Management, Evolution, Selbstorganisation* page 135

[2] Fredmund Malik *Systemische Management, Evolution, Selbstorganisation* page 140

[3] Fredmund Malik *Systemische Management, Evolution, Selbstorganisation* page 139

[4] Fredmund Malik *Systemische Management, Evolution, Selbstorganisation* page 146

System Identification based on Variable Selection and Genetic Programming III

Werner Groißböck
Fuzzy Logic Laboratorium Linz-Hagenberg
e-mail werner.groissboeck@jku.at

Johannes Kepler Universität Linz

Institut für Algebra, Stochastik und
wissensbasierte mathematische Systeme
Johannes Kepler Universität Linz
A-4040 Linz
Austria

Fuzzy Logic Laboratorium Linz-Hagenberg

FLLL
Softwarepark Hagenberg
Hauptstrasse 99
A-4232 Hagenberg
Austria

1 What is the task of system identification?

Imagine the following situation: A dynamic system is observed, and a measurement tool is available to produce a huge amount of high dimensional dynamic data, as it is the case for test bench systems in the car industry. So we have time series data with about 100 variables. And we want to be able to forecast the future behavior of one chosen variable - we call it y. In a first approach, we are content, if we can predict the next state of y, with the use of only past values of y and all the other variables. So, what we want is a formula like the following:

$$\hat{y}(t) = \beta_0 + \beta_1 \cdot y(t-1) + \beta_2 \cdot x_{25}(t-3) \cdot x_{99}(t-20) + \beta_3 \cdot \sin(x_2(t-1)) \tag{1}$$

2 Why do we use variable selection methods?

The first idea to get a good approximation formula could be to perform a linear regression algorithm with *all* of the variables. But it is well known that the quality of a formula with many degrees of freedom is bad. This effect is called 'Curse of Dimensionality'. See [6] section 7.6.1 et al. for more details. So we need algorithms that find out the most important regressor variables such that the approximation quality of the resulting formula is as high as possible when the number of degrees of freedom is restricted. But this is exactly what a variable selection algorithm can do. Variable selection algorithms take all the regressor candidates that *may* be in the approximation formula and find the most important ones.

Another reason that makes it necessary to use variable selection methods is that the calculation time would also explode, if a formula with all the variables is used.

3 What about nonlinear terms?

Variable selection algorithms in their pure form get exactly the same data as it is needed for linear regression algorithms, and they can only find a formula that is a linear combination of some of the regressors. If we want nonlinear terms too, then one very efficient way is to expand the set of regressors by artificial nonlinear terms. We know that in physical laws for example product terms play a major role. So a first approach could be to generate all the products of the original regressor candidates. This has been implemented. The problem is: If you have 100 original regressors, then the number of products is 5050. So evaluating all these product terms can be quite time consuming. If we want to get an approximation formula for dynamic systems, then theoretically some terms can be used with time shifts. So if you do not only take all product terms, but also combinations of product terms and time shifts - one of these terms might be

$$x_{25}(t-3) \cdot x_{99}(t-20)$$

then we can see that the number of possibilities explodes rapidly, and it is extremely time consuming to calculate the set of all artificial regressors in advance.

So a different approach has been tried out:

We use the Genetic Programming approach (see [1]) to generate very simple and small expansion terms. And we use a special variable selection routine that is able to calculate the quality of a new expansion term by simply calculating the correlation coefficient. And if this correlation coefficient is a hopeless candidate, then the corresponding expansion term is thrown away immediately. Important: Very often a new expansion term can be detected as hopeless, when only a few data points are evaluated. So in this case - and this is very often so, because a Genetic Programming approach produces very often an individual that is far away from being optimal - the calculation time is minimized vastly.

Furthermore, this approach has the advantage that the structure of the expansion terms that are tried out is not so restricted. There are restrictions, too, but they are much more flexible.

4 Which is our favorite variable selection algorithm?

In our previous work many variable selection routines have been tried out (see [5]). At the moment, one special variant of *forward selection* is our favorite, because it is extremely fast and still not much worse than forward selection with orthogonalization - see [6] and [2]. In our application it is not so important that the variable selection routine alone produces great but time consuming results. Instead, we prefer an algorithm that is very fast and can be combined easily with the GP-Approach to find new useful expansion terms.

The pure variable selection algorithm can be formulated in the following way:

- Calculate the correlation coefficient C with the independent y and every regressor candidate (out of n).
- Find the most important one (by comparing C). We call it x_A.
- Transform the independent variable i.e. the part of y that can be approximated linearly via x_A and the constant variable $\vec{c} = (1, \ldots, 1)^T$ is subtracted from y.
- Calculate the correlation coefficient C of y with every remaining regressor candidate (out of $n - 1$).
- Find the most important one. We call it x_B.
- Transform the independent variable y by subtracting from y the linear best approximation via the three variables x_A, x_B and $\vec{c}$.
- Continue in this manner, until a given number n_s of regressors is selected.

This algorithm can be speeded up enormously by stopping the calculation of the correlation coefficient early, if it can be seen, that the regressor is a hopeless candidate.

5 A hybrid approach based on GP, variable selection methods and stopping early

If you want to construct an approximation formula which is linear in the parameters (like in (1)) and you want to find out which nonlinear terms lead to the best approximation quality, then the following approach is very efficient:

- Use the Genetic Programming approach (see [1]) at least to generate a random population of very simple terms (to get additional regressors in every step) and then use the described variable selection algorithm to find out the 'most important' terms by simply calculating correlation coefficients.
- Usually in Genetic Programming, many hopeless individuals appear. So stop the calculation of the correlation coefficient and even the evaluation of the individual early, if it is hopeless.

With these ideas, the new hybrid algorithm can be reformulated as follows:

1. Generate a new generation as usual in the Genetic Programming approach.
2. Calculate the correlation coefficient C with the independent y and every individual of the population.
3. If it can be seen that an individual is hopeless, then the GP-function-evaluation and the calculation of the correlation coefficient can be stopped early to save time.
4. Find the most important one (by comparing C). We call it x_A.
5. Transform the independent variable i.e. the part of y that can be approximated linearly via x_A and the constant variable $\vec{c} = (1, \ \dots \ , 1)^T$ is subtracted from y.
6. Generate a new generation as usual in the Genetic Programming approach.
7. Calculate the correlation coefficient C with the independent y and every individual of the population.
8. If it can be seen that an individual is hopeless, then the GP-function-evaluation and the calculation of the correlation coefficient can be stopped early to save time.
9. Find the most important one. We call it x_B.
10. Transform the independent variable y by subtracting from y the linear best approximation via the three variables x_A, x_B and $\vec{c}$.
11. Continue in this manner, until a given number n_s of regressors is selected.

This algorithm is a very promising approach, but there are still a lot of possibilities to improve the performance.

6 First experimental results

Till now, two very basic variants have been implemented. Both variants do not use genetic operators. The only thing they do is that they generate a huge initial population of formulas as usual in Genetic Programming. And they simply look for the best individual in this population by calculating the correlation coefficient with the actual independent y. Our first variant does exactly this. The second variant is very similar, but the stopping-early-idea has been implemented. At the moment, for only 30 rows of the data set (which consists of hundreds of rows) the actual individual is evaluated and the correlation coefficient is evaluated. And if the actual individual is not approximately as much correlated as the best individual so far, then the actual individual is thrown away immediately. It has to be mentioned that before the second variant is applied, the data rows are permuted. This has to be done because we do not want that correlations that appear only in later measurements are neglected.

To compare the two variants in a fair way, every variant is allowed to try out new individuals, until a given time is reached.

The first experiments have been performed with 20 seconds for finding *one* expansion term. We noticed, that performance of the two variants is nearly the same. But then we made the same experiments with only 1 second, and we noticed, that the algorithm with stopping early produces significantly better results than the other.

References

[1] J. R. Koza, "Genetic Programming", ISBN 0262111705 The MIT Press, Cambridge, Massachusetts 1992

[2] A. Miller, "Subset Selection in Regression - Second Edition", ISBN 1-58488-171-2 Chapman & Hall/CRC Boca Raton London New York Washington, D.C. 2002

[3] F. E. Harrell jr., "Regression modeling strategies: With applications to linear models, logistic regression and survival analysis", Springer Series in Statistics

[4] L. Ljung, "System Identification: Theory for the User.", ISBN 0-13-656695-2, Prentice Hall PTR, New Jersey, 1999

[5] W. Groißböck, E. Lughofer "A Theoretical and Practical Comparison of Variable Selection Methods with the Main Focus on Orthogonalization", *Technical Report FLLL-TR-0211*

[6] O. Nelles, "Nonlinear System Identification - From Classical Approaches to Neural Networks and Fuzzy Models", ISBN 3-540-67369-5 Springer-Verlag Berlin Heidelberg New York

[7] M. R. Spiegel, "Statistik", McGraw-Hill Book Company Europe

[8] L. Xu and W. Zhang, "Comparison of different methods for variable selection", Analytica Chimica Acta, Volume 446, Issues 1-2, 19 November 2001, Pages 475-481.

s c c h
software competence center
hagenberg

Print Layer Identification in Series of Multi Layer Print Images

Daniel Soukup
Daniel.Soukup@scch.at
May 7, 2004

Abstract — The topic of the talk is to identify the print layers in series of multi layer print images. Feature points are detected in each of the sample images. These feature points are matched between the sample images and the sample images are aligned according to the transformation parameters obtained by the point pattern matching process. Arithmetic difference images of the brightness values of the aligned images are computed and the maximum of the brightness values of all these difference images is taken. For one tupel of transformation parameters, i.e. one print layer, an image is obtained, in which a dark region indicates the searched print layer and regions of all other print layers are entirely bright.

Key words — *print layer identification,multi layer prints, feature point detection, point pattern matching, brightness difference images*

Kplus
Kompetenzzentren-Programm

1 Problem Description

The problem arises in multi layer print processes: different regions of an image are printed in separate steps of work. Each such region is called a layer. Such a region separation is determined by different colors or different print methods of the image parts. The print machines have to be calibrated precisely, so that the print layers are correctly arranged. It could happen that some print layers are incorrectly shifted with respect to each other. Figure1 shows such an error. The red ellipse has an unwanted overlap with the blue rectangle on one side, while the gap on the other side is much too wide. Some of such shifts may be tolerable while others are too severe.

In practice a few images are printed after precalibration and a digital image of such test prints is investigated by a human print inspector, who decides, if the outcome is sufficiently precise and recalibrates the print machines if not. He masks each region of the print layers and looks, if each such region is correctly positioned with respect to regions of other layers especially neighboring ones. In figure1 we can easily recognize that such an image probably consists of three print layers (figure2). For complicated prints these regions are not easily identified, which makes this process cost a lot of time.

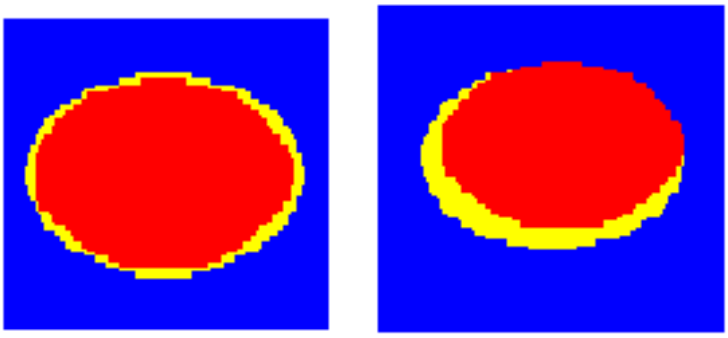


Figure 1: Print Error

The goal of this work is to develop methods to make this error identification process faster, so that the error detection time and thus the calibration time becomes as short as possible.

The problem is to identify the different print layers. A sufficiently large set of sample images is given. Within the work it has exactly to be found out how many samples are needed to make the set sufficiently large. Out of this sample set those regions have to be found, whose member pixels underlie common affine transformations from each sample to any other. The identification will be realized in an offline process. It is not important to identify a whole print layer as one, it is enough to identify each part of a print layer as a connected region.

Figure 2: Identified Print Layers

2 Idea for Solution

First one image of the sample image set is fixed as initial image and we try to align each of the remaining images to the fixed one for every layer. For this step we have to find feature points in the sample images. I have chosen to search for corner points. Therefor the Soyka Corner Detector [Soyka] is used.

Then a point pattern matching is applied between the fixed image's feature points and those of all the other images. The difficulty hereby is that not all the corners of one image do match all the corners of a second one with only one tuple of transformation parameters. As the different layers are shifted with respect to each other in different ways, their feature points are so, too. For every print layer the transformation parameter tuple will be different. How this could be managed is mentioned in the next section.

As a result we obtain the matching points and their corresponding transformation parameters. The transformation parameters then have to be grouped according to their similarity. Those corresponding feature point pairs that underlie the same transformation parameters will belong to the same print layer.

In the following we only concentrate on finding one print layer region, the others are found in the same way. Each sample image is aligned to the fixed image so, that the feature points of the current layer cover their correspondents in the fixed image.

Then for this alignment the absolute arithmetic difference of the brightness values of each image pixel pair that overlaps for the current alignment is computed. The result will be that the image part that contains the aligned feature points will be entirely dark, because this region is matched exactly. In the remaining image parts many white pixels will arise, because for these parts the matching is not correct, the differences will be high (figure 4 and 5). Of course, for dense drawings, there will be many other dark points, which are not part of the current print layer. Their brightness has just been erased coincidently. To overcome these coincidental matchings the procedure is repeated between each sample image and the fixed one. Thus a set of difference images is obtained. Each difference image is dark in the region of the searched print layer. At last the maximum of the brightness values of all these difference images is computed. As a result we obtain an image that has a dark spot only in the print layer region. The high difference values in the other image parts that are differently positioned in each of the difference images will have been blurred over the entire image area except the current layer part. The print layer is found in the dark image part and as all the transformation parameter values have been stored within the point

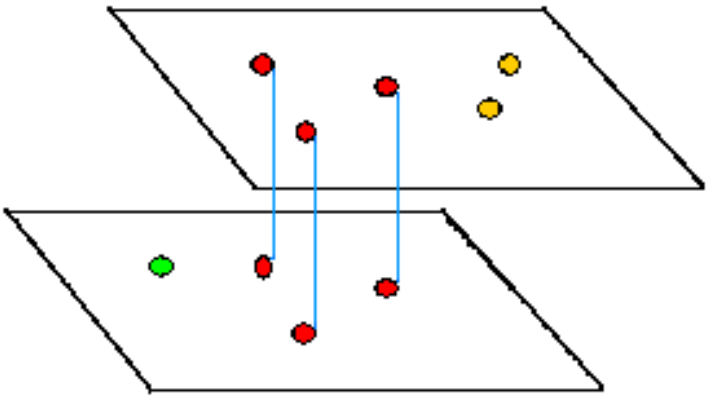


Figure 3: Image Alignment

pattern matching process the print layer is located in the initially fixed image as well as in all the other sample images.

Figure 4: Three geometric objects are composed. They are shifted in different directions with respect to each other form the left image to the right

3 Point Pattern Matching

The point pattern matching algorithm that will be applied [PP] can handle translation, rotation and scaling differences between two point sets $P = \{p_i : i = 1, ..., n\}$ and $Q = \{q_a : a = 1, ..., m\}$, and is able to handle extra or missing points condition. The last ability will be important in our problem, that the two feature point sets cannot be matched as a whole.

The algorithm first computes the Matching Pairs Support for each possible matching pair (p_i, q_a), i.e. the number of other possible matching pairs (p_j, q_b) that support the matching between p_i and q_a. (p_j, q_b) is said to support the matching of the pair (p_i, q_a), if the vectors $\overline{p_i p_j}$ and $\overline{q_a q_b}$ are parallel and have an equal length (figure 6). This procedure is applied as follows:

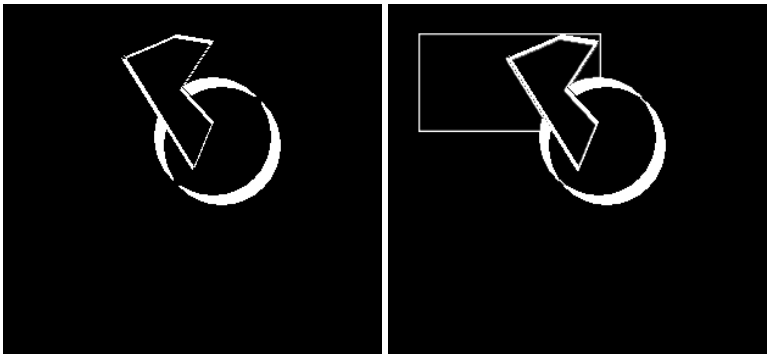


Figure 5: Difference images of the relatively shifted geometric object composition images. For difference image 1 the rectangles are perfectly matched

1. A possible matching pair p_i and q_a is considered to be a correct match
2. For all other possible matching pairs p_j and q_b the angle Θ between $\overline{p_i p_j}$ and $\overline{q_a q_b}$ and the scaling factor $s = \frac{\overline{q_a q_b}}{\overline{p_i p_j}}$ are computed
3. For these values an accumulator array is incremented at position (Θ, s). The accumulator array accumulates the angles and scaling factors of the investigated vectors
4. Find the maximum w_{ia} in the accumulator array and store it and the corresponding transformation parameters in a Matching Point Support Matrix at position $MPS(i, a)$

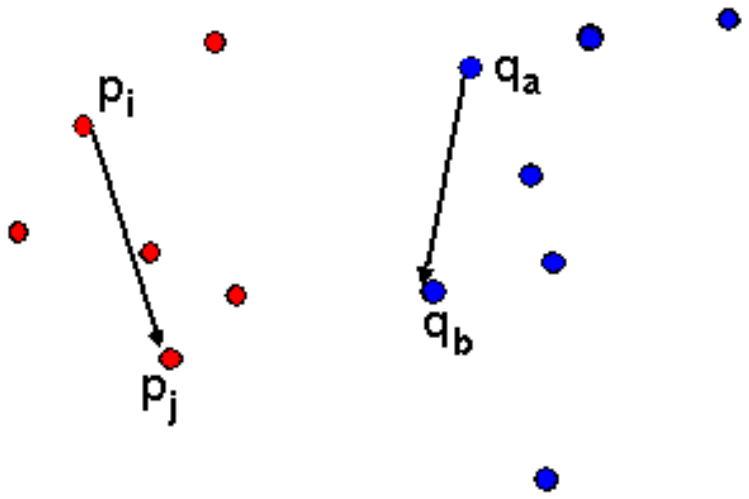


Figure 6: The difference vectors $\overline{p_i p_j}$ and $\overline{q_a q_b}$

This procedure is repeated for all possible matching pairs p_i and q_a. Now we have to state when a matching between two points is "possible": the computational expense of this algorithm

is very high. Usually all pairs are possible matches, but in our special case of comparing print images, we can assume that a pair is only a possible matching pair, if the two points are members of the same small neighborhood. The size of this neighborhood still has to be explored, but I estimate it smaller than 7 pixel.

The presented algorithm provides the MPS Matrix. In this matrix one searches for all entries that are equal, i.e. entries that have the same weight w_{ia} and the same assigned transformation parameters. All points with approximately the same entries in the MPS Matrix are feature points of the same print layer.

References

[PP] SHIH-HSU CHANG, FANG-HSUAN CHENG, WEN-HSING HSU, GUO-ZUA WU *Fast Algorithm for point pattern matching: invariant to translations, rotations and scale changes*,Pattern Recognition, Vol. 30, No. 2, pp. 311-320, 1997

[Soyka] Eduard Soyka *Corner Detection Algorithm*, VSB-Technical University of Ostrava, 2003, http://www.cs.vsb.cz/sojka/cordet/presentation.html

Incremental version of the IPAN method

Leila Muresan
Fuzzy Logic Laboratorium Linz
e-mail: leila.muresan@jku.at

Introduction

Tracking of independent point-like features combines two basic steps: detection and data association (trajectory construction). In the special case of single molecule tracking the quality of detection is crucial for the quality of tracking, given the following two reasons:

1. false positives and false negatives occur with high probability in the case of weak signals on noisy backgrounds
2. Brownian motion is assumed , meaning that no characteristics of the movement (e.g. smoothness) can be used for tracking

Since finding the exact detection threshold is practically impossible, either the uncertainty of detection penalizes the associations (it is incorporated in the cost function) or trajectories are constructed incrementally, by adding one particle at a time, (the particles being ordered according to the reliability of detection). The latter improves the solutions affected by false negatives; however it doesn't improve the case of false positives (but offers additional information, so that special techniques might be implemented for their detection).

Linear assignment problem

A first approach to the tracking problem is to consider particles in successive frames and to assign pairs of particles as to solve the linear assignment problem (see [3]):

$$\min \sum_i \sum_j c_{ij} x_{ij}$$

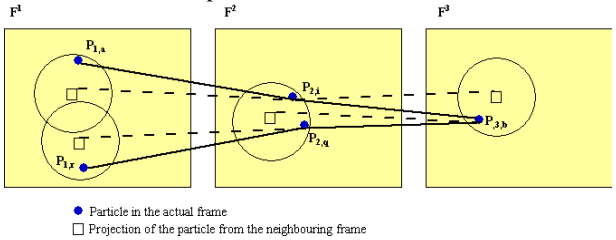
$$\sum_i x_{ij} = 1$$

$$\sum_j x_{ij} = 1$$

The solution (obtained by Hungarian method, for example or some approximation as in [1]) is extremely sensitive to misdetections. One single misdetection may destroy all the associations in the matching. Therefore, we need an algorithm for which the effect of misdetection is local.

IPAN

A more robust algorithm is the IPAN method, presented in [2].



Initialization:

For all i particles in F^2:

1. Project the location of P_i^2 is on F^3 and F^1,
2. Determine S_i^{2+} and S_i^{2-} - search areas – gates - of i
   (The gate of i is an area, represented by a disk, that has a center in the position of i projected on the predecessor or successor frame, and a radius radius r, which is a threshold of the movement of the particle between two frames)

3. From all the triplets $\left(P_u^1, P_i^2, P_v^3\right)$, choose $\left(P_a^1, P_i^2, P_b^3\right)$ such that $P_q^2 \in S_b^{3-} : c(P_r^1, P_q^2, P_b^3) < c(P_a^1, P_i^2, P_b^3)$ (c is an appropriately chosen cost function-described above).
4. $\left(P_a^1, P_i^2, P_b^3\right)$ the first trajectory hypothesis
   Rank the other triples according to their cost values.
5. Test the first hypothesis:
   If $P_q^2 \in S_b^{3-} : c(P_r^1, P_q^2, P_b^3) < c(P_a^1, P_i^2, P_b^3)$ then
   the first hypothesis is rejected and replaced with the second ranking hypothesis.
   Else
   If $P_q^2 \in S_b^{3-} : c(P_r^1, P_q^2, P_b^3) < c(P_a^1, P_i^2, P_b^3)$ then
   the first hypothesis is rejected and replaced with the second ranking hypothesis.
   Same test for P_a^1.
6. If the hypothesis is not rejected, the triple $\left(P_a^1, P_i^2, P_b^3\right)$ is the beginning of a particle trajectory. If all hypothesis for P_i^2 have been rejected, the point remains unlinked.

For the **subsequent frames** the same method is applied. Note that at time *k*, in frame F^{k+1} all the points are unmatched, in frame F^k some points are isolated, other are linked with points in the previous frame, while in frame F^{k-1} we can have isolated, double-linked or single linked points too.

Towards an incremental algorithm

We denote by $D^{k-1,k}$ the squared distance matrix of particles from frame *k*-1 and those from frame *k*, respecting the gate condition (when bigger, it equals to ∞). Then
$D^{l-1,l,l+1}[i,k] \overset{\Delta}{=} D^{l-1,l} \otimes D^{l,l+1}[i,k] = \min_j \{D^{l-1,l}[i,j] + D^{l,l+1}[j,k]\}$ is the matrix of shortest paths from node *i* in frame *k*-1 to node *k* in frame *k*+1.

It is easy to see that the **initialization step** becomes:
A triplet (i_0, j_0, k_0), $i_0 \in F^{l-1}$, $j_0 \in F^l$, $k_0 \in F^{l+1}$, is part of a trajectory iff

1. $i_0 = \arg\min_i D^{l-1,l,l+1}[i, j_0]$
2. $k_0 = \arg\min_k D^{l-1,l,l+1}[i_0, k]$
3. $(\neg\exists)(i^*, j_0, k^*)$ $D^{l-1,l,l+1}[i^*, k^*] = D^{l-1,l}[i^*, j_0] + D^{l,l+1}[j_0, k^*] < D^{l-1,l}[i_0, j_0] + D^{l,l+1}[j_0, k_0]$

At least one such triplet exists: for $\min_i \min_k D^{l-1,l,l+1}[i,k]$, with j appropriately chosen, all conditions are satisfied.

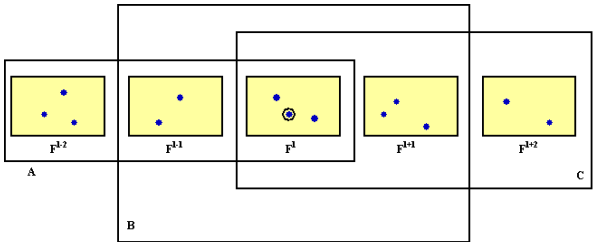
The worst case complexity of the algorithm is: O(n^3).

- Computing $D^{l-1,l} \otimes D^{l,l+1}[i,k]$ (and remembering the adequate *j*'s):- at most O(n^3)
- Finding the right triplets: 2*n*
- Removing duplicate *j*'s: at most *n*-1.

The **subsequent frames step** differs from the initialization one only in the construction of the matrix $D^{l-1,l,l+1}$: if (i_0, j_0) belongs already to a trajectory, all the elements in the row i_0 and column j_0 of $D^{l-1,l}$ are replaced with ∞ except on the position (i_0, j_0). In this case the first condition can be dropped.

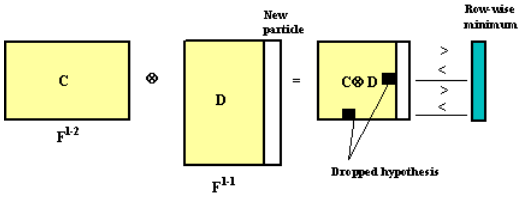
Incremental version:

Adding one particle to a frame influences at most two preceding and two succeeding frames.



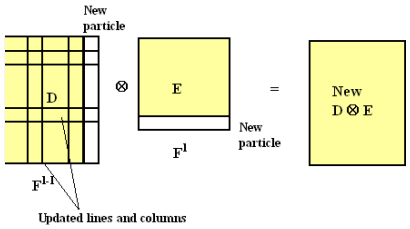
The influence of one added particle

Step A:
The new particle may destroy some old hypothesis (breaks the last), and can create at most one new one.



If the new $C \otimes D$, with the additional column generated by the newly added particle, has size $n_1 \times n_3$, the dropped hypothesis are those for which dropped hypotheses are the ones for which:
$C \otimes D[i, n_3] < \min_{j=1,\ldots n_3-1} C \otimes D[i, j]$. To compute the last column of $C \otimes D$, O(n^2) additions are needed.
For every broken association (*j*,*k*) in *D* the ∞ in line *j* and column *k* are replaced with the original (gated) distances.

Step B:



$$New \quad D \otimes E[p,q] = \begin{cases} \min_s \{D[p,s] + E[s,q]\}, \\ \min\{Old\ D \otimes E[p,q], \min_s \{D[p,s] + E[s,q] \mid s \in Dropped \cup \{n_3\}\}, \\ Old\ D \otimes E[p,q], otherwise \end{cases}$$

where $Dropped = \{k \mid \exists (j,k)$ dropped in step A$\}$.

The 2*n* steps of finding triplets (new hypothesis NH) are performed. If an old hypothesis is not in the NH set, E is updated accordingly.

Step C:
Similar to B and A.

Remark:
The fact that a new particle destroys many trajectories and doesn't construct any hypothesis, might be a hint to exclude it from the final solution.
A particle that destroys in a late phase (meaning a low detection threshold) a very early hypothesis, might be ignored.

The time complexity of the incremental version is comparable with the time complexity of the first-detect-then-track approach. However, a drawback of the former method is the increased spatial complexity (The distance matrices cannot be discarded as in the static case, moreover "product matrices" are also allocated and updated,

during the steps of the algorithm). If the space requirement becomes too large, the movie can be segmented in shorter sequences.

Further work

Reasonable stopping conditions are still to be found, problem closely related to quantifying the "goodness" of a trajectory. The influence of the particle position's precision, as well as the influence of the gate radius still has to be studied.
Also there seem to exist promising connections to the (max, +) algebra, to the dynamic shortest path method and the transshipment problem.

References

1. Blackman S., Popoli R. – "Design and Analysis of Modern Tracking Systems", Artech House, 1999
2. Chetverikov D., Verestoy J. –"Feature Point Tracking for Incomplete Trajectories", Computing, Devoted Issue on Digital Image Processing, vol.62, pp.321-338, 1999
3. Muresan, L. – "Tracking in microscopy images", Master and PhD Seminar, November 2003