Scalable Recursive Top-Down Hierarchical Clustering Approach with implicit Model Selection for Textual Data Sets

Markus Muhr
Knowledge Relationship Discovery
Know-Center Graz
Graz, Austria
mmuhr@know-center.at

Vedran Sabol
Knowledge Relationship Discovery
Know-Center Graz
Graz, Austria
vsabol@know-center.at

Michael Granitzer
Institute of Knowledge Management
Know-Center Graz
Graz University of Technology
Graz, Austria
mgranitzer@tugraz.at

***Abstract*—Automatic generation of taxonomies can be useful for a wide area of applications. In our application scenario a topical hierarchy should be constructed reasonably fast from a large document collection to aid browsing of the data set. The hierarchy should also be used by the InfoSky projection algorithm to create an information landscape visualization suitable for explorative navigation of the data. We developed an algorithm that applies a scalable, recursive, top-down clustering approach to generate a dynamic concept hierarchy. The algorithm recursively applies a workflow consisting of preprocessing, clustering, cluster labeling and projection into 2D space. Besides presenting and discussing the benefits of combining hierarchy browsing with visual exploration, we also investigate the clustering results achieved on a real world data set.**

***Keywords*-topic hierarchy; landscape; growing k-means; model selection; vector space model;**

I. Introduction

Explorative navigation is a common approach for users who need to gain insight into large document collections. Different application scenarios, such as browsing of document repositories or organizing search results, benefit from automatically created topic hierarchies or concept hierarchies. We propose an approach build upon a scalable, recursive clustering algorithm, which is applied on document sets to generate a topical hierarchy suitable for browsing. The hierarchy is supplemented with a visual navigation and exploration aid based on the information landscape visual metaphor.

Kummamaru et. al. [1] made a survey of various approaches for automatic taxonomy generation. These methods can be categorized into monothetic (clustering based on single features) and polythetic (clustering using multiple features) algorithms, as well as into top-down (partitioning) and bottom-up (agglomeration) approaches. In the survey various algorithms are divided into methods based on clustering of documents, clustering of words. and clustering documents and words simultaneously (also called co-clustering).

Cutting et. al. [2] introduced an approach called Gather/Scatter to aid the user in information retrieval tasks. It basically clusters search results to provide the user with a partitioning depending on concepts present in the search result set. The user can iteratively refine the search results by selecting interesting clusters and rerun the clustering to obtain a new, more detailed partitioning of the selected search results. Due to favorable user feedback there is still research going on to improve this idea, like for example by [3].

Andrews et. al. [4] showed that the InfoSky visualization technique can aid the user in exploring hierarchically organized document collections. The visualization is basically a 2D layout of classes and documents where spatial proximity is a measure for their topical relatedness. The algorithm employed to generate the layout requires that a hierarchy is specified externally. However, such pre-defined hierarchies are usually not available, especially since manually crafted hierarchies are rare due to the labor-intensive task required for creating them. For this reason, an automatic generation of a topic hierarchy appears to be a natural extension of this concept. Sabol et. al. [5] outlined such an approach, which will be described in detail from a technical point in the following.

Since Gather/Scatter and InfoSky proved their usefulness in aiding the user to explore and navigate document collections, we envisioned an algorithm for generating a hierarchy that can be used both for browsing of large document collections as well as for explorative visualization using the InfoSky approach. The following feature list defines key aspects of such an algorithm:

- Hierarchical, top-down, polythetic, document clustering approach, into which the InfoSky projection procedure can be directly embedded.
- Dynamic cluster structure on each level of the hierarchy supporting splitting and merging of clusters.
- Constraints on the maximum and minimum number of elements per hierarchy level due to usability considerations for hierarchy browsing. Also, this keeps the computational costs of the InfoSky projection algorithm within well-defined bounds.
- Scalable to data sets consisting of millions of documents with a reasonable trade-off between runtime and

accuracy.
- Reasonable cluster labeling to support browsing and navigation.

To achieve these constraints we introduce a top-down, recursive, scalable clustering algorithm for creating a topical hierarchy from a document collection. This algorithm generates the hierarchy by recursively executing a workflow consisting of preprocessing, flat (i.e. partitional) clustering with dynamic number of clusters, cluster labeling and projection. This workflow is first applied on the whole data set, and then recursively reapplied along the generated clusters on ever smaller data subsets. The recursion stops at clusters with document count sufficiently small for direct inspection by the user.

The preprocessing part consists of feature weighting, selection and normalization [6]. As flat clustering algorithm we use growing k-means [7] which is based on clustering methods such as Neural Gas (generalization of online k-means) and Growing Neural Gas [8], and which, like for example the spherical batch k-means introduced by Dhillon et. al. [9], optimizes the cosine similarity. We have chosen growing k-means due to its good trade-off between performance (linear runtime) and reasonably good accuracy [10], as opposed to algorithms such as DBScan or Chameleon which are of quadratic time complexity in their basic form, and algorithms like BIRCH which is excessively order-dependent and inaccurate. To reduce the running time even further we used the heuristic optimizations introduced in [11]. Another advantage of growing k-means is that it offers a very efficient way of performing model selection. Due to its iteratively increasing number of clusters, one can naturally measure the fitness of the clustering for different number of clusters. Fitness can be evaluated by internal validity indices like Bayesian Information Criterion (BIC) [12] or stability-based approaches like the stability method [13]. On the fitness curve obtained for different number of clusters (within the permitted maximum and minimum), we detect the knee point (or sharp point) to identify the best fitting model by utilizing *DiffKnee* [12]. Cluster Labeling is performed with Jensen-Shannon divergence [14]. Finally, the projection procedure is basically the same as outlined for the InfoSky algorithm [4].

The rest of the paper is organized as follows: In section II an overview approach of our hierarchical clustering algorithm is given. In section III the main workflow components (preprocessing, clustering, labeling, and projection) are described into detail. The experiments section IV provides a presentation and discussion of the advantages that such an automatically generated taxonomy offers for browsing and visualizing document collections. Furthermore, an evaluation on splits of the Wikipedia used at INEX 2009 is provided. Finally, in section V we provide a summary and an outlook on future work.

II. Top-Down Recursive Hierarchical Clustering Algorithm

For the purpose of this paper we define a concept hierarchy as a tree structure where the leaves represent documents, while all higher level nodes in the tree are clusters/concepts. Our method incorporates a simple recursive, partitional, top-down clustering approach with constraints on the minimum and maximum child number at each hierarchy level (which is the same for sub-clusters and documents). These constraints limit the branching factor for each cluster node for two reasons: to guarantee a certain runtime behavior of the InfoSky projection procedure as well as to provide a comfortable and usable browsing experience for the user. In our method arbitrary algorithms for document preprocessing, clustering and labeling may be used. In a final step positioning of clusters and documents on a 2-dimensional space can be performed using the InfoSky algorithm.

The algorithm follows a divide and conquer schema. The initialization includes passing the input data set and the constraints on the minimum and maximum number of nodes on each level. The whole input data set, together with the root node of the hierarchy, is pushed onto the list of remaining tasks. Tasks are popped from the list and executed, which generates further sub-tasks. Because tasks are independent, several can be popped and executed simultaneously in the case there are more processing cores available. The algorithm terminates when no more task are executing and the task list is empty. Processing of each task consists of the following steps:

Step 1: Preprocess the documents belonging to the task. The preprocessing transforms the documents in a form appropriate for executing the following steps.

Step 2: Cluster the documents belonging to the task using an arbitrary flat clustering algorithm. The clustering creates a list of clusters with documents attached to clusters as direct children.

Step 3: Run splitting and merging of clusters to fulfill the following constraints:

- The number of clusters must be between the given constraints. If there are fewer clusters than the minimum constraint requires, clusters are split (biggest or least coherent cluster). If there are more cluster than the maximum allowed number, clusters are merged (two most similar clusters).
- The number of documents belonging to each cluster must be between the provided limits, or be at least 1.5 times the upper limit so that the degree of freedom for a new recursive level is large enough for meaningful clustering. If these conditions are not met, cluster splitting and merging is performed until they are satisfied.

Step 4: Clusters having the number of documents between the required minimum and maximum are considered completed and will not be clustered recursively. All other clusters should have a number of documents at least 1.5 times the upper limit. These are pushed on the task list to create new recursive levels of sub-clusters.

Step 5: Label the clusters.

Step 6: Project the clusters into a 2 dimensional space by using the InfoSky algorithm. For clusters which are completed also project the documents.

III. Hierarchy Level Generation

In this section we will describe the algorithms used to process a task.

A. Preprocessing

The preprocessing includes weighting, followed by feature selection and normalization. For weighting we use the well-known BM-25 schema [6]. For feature selection we utilize quantil pruning, where those features of a document vector are removed which fall below a given quantil of the l1-norm of the vector. Finally, the vectors are normalized to unit length effectively projecting them onto a hypersphere.

B. Clustering

For high-dimensional data like text documents the direction of a vector is more important than its actual length. For this reason, the most common cost function for a k-means on the unit-sphere (spherical k-means) tries to maximize the cosine similarity between the data samples and their most similar centroid.

One variant of a spherical k-means is the growing k-means (algorithm 1). Given a data set $\mathcal{X}$ and a number of clusters K it outputs centroids $\mathcal{C}$ with assignments of data samples to the clusters $\mathcal{Y}$. The algorithm starts with 2 initial centroids and incrementally adds new centroids until K clusters are created. For each number of clusters two loops over the data set are performed. The first one to find the nearest centroid for each sample and to update this centroid utilizing an online adaption with a decreasing learning rate ($c_{y_n} = c_{y_n} + \eta x_n$). The second loop calculates the current cluster assignments $\mathcal{Y}$ and the average similarity of the data samples to their centroids (s_k). Finally, the centroid with the least average similarity to its data samples is identified, and in the middle between this centroid and its most dissimilar data sample a new node (i.e. cluster centroid) is inserted.

Algorithm 1 Growing Spherical K-Means

```
input:
$\mathcal{X} = \{x_1, \dots, x_N\}$ with $x_i \in \Re^d$, $K$, $l$, $\eta$, $\nu$
output:
$\mathcal{C} = \{c_1, \dots, c_K\}$, $\mathcal{Y} = \{y_1, \dots, y_N\}$ $\forall$ $y_n \in \{1, \dots, K\}$
steps:
initialize centroids $c_1$ and $c_2$ by a seeding mechanism
for $m = 2$ to $K$ do
    for $n = 1$ to $N$ do
        $y_p = y_n$
        $y_n = \arg\max_{1 \le k \le m} x_n^T c_k$
        $c_{y_n} = c_{y_n} + \eta x_n$
        $c_{y_p} = c_{y_p} - \nu x_n$
        if $||c_{y_n}|| - 1.0 > l$ then
            $c_{y_n} = \frac{c_{y_n}}{||c_{y_n}||}$
    for $n = 1$ to $N$ do
        $y_n = \arg\max_{1 \le k \le m} x_n^T c_k$
        $s_k = s_k + \max_{1 \le k \le m} x_n^T c_k$
    if $m < K$ then
        $c_i = \arg\min_{1 \le k \le m} S(c_k)$
        $x_j = \arg\min_{x \in \mathcal{X}_i} x^T c_i$ with $\mathcal{X}_i = \{x_n | y_n = i\}$
        $c_t = \frac{c_i - x_j}{2}$, $\mathcal{C} = \mathcal{C} \cup \{c_t\}$
```

A considerable reduction in effective runtime is achieved by using a simplified update function compared to the geometrically correct update of $c_{y_n} = \frac{c_{y_n} + \eta(x_n - c_{y_n})}{||c_{y_n} + \eta(x_n - c_{y_n})||}$, as well as by employing a heuristic that postpones the normalization on the cluster centroid until the length of the vector gets too long ($||c_{y_n}|| - 1.0 > l$). Both heuristics, introduced in Zhong [11], are valid due to the spherical nature of the algorithm. As learning rate η we utilized one that decreases with the square root of the cluster size ($\eta = 1/|\sqrt{\mathcal{X}_{k(x)}}|$). We additionally introduce a repulsion term which pushes the old centroid a little bit away from the document, so that a faster and better separation between the old and the new centroid can be achieved ($c_{y_p} = c_{y_p} - \nu x_n$).

As seeding mechanism for the initial two clusters of the growing k-means a directed randomized method, also known as k-means++ method, is used [15]. The first seed is taken randomly from the data set and the others are selected to maximize the dissimilarity to already selected seeds.

We utilize model selection to find a suitable number of clusters. Growing k-means provides intermediate results for each cluster configuration starting with the one consisting of 2 clusters (one could also start with a user-defined number of clusters), whereby the fitness of the current result is measured before each consecutive insertion. In this way model selection can be performed without having to execute the clustering algorithm anew for different number of clusters.

We utilize two different fitness indices. The first one is the stability method [13] which relates the cluster results and the prediction results to maximize the agreement between the labels. Furthermore, we also use the Bayesian Information Criterion (BIC) as statistical criterion to assess model fitness [12]. Using those two indices a fitness curve over the growing cluster number can be computed. However, the first decisive local maximum does not always deliver a satisfying cluster number selection. Zhao et. al. [12] introduced *DiffBIC*, which is a more substantiated method to find the sharp point or knee point of such a fitness curve. We apply the method to both indices and call it therefore *DiffIndex*. *DiffIndex* provides us with the best number of clusters and if the intermediate cluster results have been stored, will immediately output the final clustering result.

C. Cluster Labeling

Glover et. al. [14] showed that when external knowledge is not available Jensen-Shannon divergence seems to provide very good labeling results. We apply this method for cluster labeling, however, it should be noted that we did not perform a detailed evaluation.

D. InfoSky Projection

InfoSky projections algorithm computes a 2D representation of the cluster hierarchy such that high-dimensional similarity is represented by spatial proximity in 2D. It was designed for efficiently projecting large, hierarchically organized document collections. The algorithm proceeds recursively by positioning clusters on each level of the hierarchy using a force directed placement (FDP) algorithm, which has a time complexity of $O(n^3)$. The resulting vertices are inscribed into a Voronoi region belonging to their parent cluster, which is then further subdivided into smaller, nested Voronoi areas. The recursion stops when it reaches the leaves (i.e. documents), which are positioned and inscribed in the same way as clusters. As long as the number of child nodes on each hierarchy level has an upper limit (which is here the case), each FDP run can be considered to execute in constant time, yielding to a complexity of roughly $O(n * log(n))$ for the whole projection algorithm. The detailed description of the InfoSky method can be found in [4].

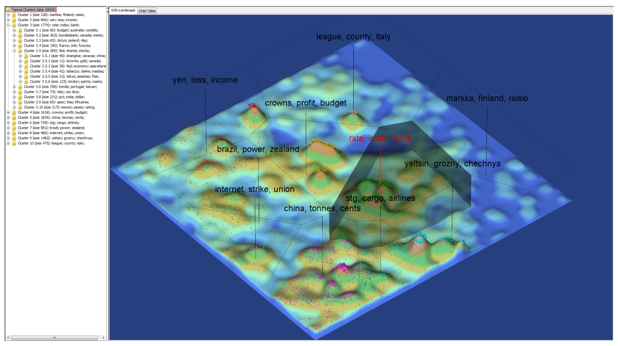
IV. Experiments

Besides discussing an application scenario involving visualization, where the hierarchy is used to drive a scalable projection algorithm, we tested our clustering algorithm on a Wikipedia split provided at INEX 2009.

A. Application Scenario: Landscape Visualization

In [5] Sabol et. al. give a detailed description of a graphical user interface including a cluster hierarchy browsing component and a landscape visualization computed by the described algorithm. In figure 1 an example of such a user interface is shown.

To assess the benefits provided by the hierarchy of topical clusters and by the similarity layout provided by the associated information landscape we performed heuristic evaluation of new functionality during the design phase. Formal experiments and thinking aloud tests were conducted at later development phases. Experiments performed in [4] suggest that the combination of an information landscape with tree and table widgets does offer advantages in use cases where the goal is to learn about topical relationships, discover topical clusters and gain an overview of the previously unexplored data. Experiments described in [5] focused on explorative browsing in multiple coordinated



- Topical Clusters (size 10000)
  - Cluster 1 (size 126): markka; finland; raisio;
  - Cluster 2 (size 906): yen; loss; income;
  - Cluster 3 (size 1774): rate; index; bank;
    - Cluster 3.1 (size 66): budget; australia; costello;
    - Cluster 3.2 (size 363): bundesbank; canada; marks;
    - Cluster 3.3 (size 65): zlotys; poland; nbp;
    - Cluster 3.4 (size 156): francs; snb; futures;
    - Cluster 3.5 (size 285): fed; shares; stocks;
      - Cluster 3.5.1 (size 40): shanghai; caracas; china;
      - Cluster 3.5.2 (size 11): toronto; gold; canada;
      - Cluster 3.5.3 (size 39): fed; economy; operations
      - Cluster 3.5.4 (size 41): tobacco; claims; nasdaq;
      - Cluster 3.5.5 (size 31): tokyo; pesetas; ftse;
      - Cluster 3.5.6 (size 123): london; points; marks;
    - Cluster 3.6 (size 258): bonds; portugal; taiwan;
    - Cluster 3.7 (size 73): italy; cpi; ibca;
    - Cluster 3.8 (size 231): pct; india; dollar;
    - Cluster 3.9 (size 60): apec; litas; lithuania;
    - Cluster 3.10 (size 217): mexico; pesos; rating;
  - Cluster 4 (size 1616): crowns; profit; budget;
  - Cluster 5 (size 1835): china; tonnes; cents;
  - Cluster 6 (size 739): stg; cargo; airlines;
  - Cluster 7 (size 581): brazil; power; zealand;
  - Cluster 8 (size 466): internet; strike; union;
  - Cluster 9 (size 1482): yeltsin; grozny; chechnya;
  - Cluster 10 (size 475): league; county; italy;

Figure 1. An example user interface showing the cluster hierarchy in a tree and in a landscape.

view environments. Automated zooming and panning to user's point of interest were found to be slightly better than purely manual navigation, however the results turned out to be less conclusive than expected.

B. Evaluation on Wikipedia splits (INEX 2009)

1) Data: For our experiment we applied the algorithm to small-scale as well as to the large-scale data of the INEX xml mining challenge 2009 [1]. The INEX XML Wikipedia data set is a split of the English Wikipedia where the documents have already been preprocessed, and are provided in a bag-of-words representation including terms and frequent phrases (specifically term vectors of uni-grams and bi-grams). The two splits of the English Wikipedia consist of 54,575 documents as well as 2,666,190 documents.

2) Methods: We used the provided online evaluation tool of the INEX challenge found at [2] to evaluate our hierarchy. Clustering results were evaluated against the category schema provided by the YAGO especially concerning micro and macro purity. Our settings for each of the methods were using BM-25 weighting and a 0.5 quantil pruning as well as a 0.8 learning rate and a square root weight for the decaying factor. Furthermore, since the evaluation does not favor hierarchical structures directly, but allowed multiple clusters for one document, we assigned each document to each cluster in its complete parent structure. In the following the online evaluation tool at INEX 2009 provided us with macro and micro purity values which are obviously dependent on the number of clusters, so that we have tried to keep the cluster number constant on different runs.

3) Results and Discussion: Our resulting hierarchy, consisting of 10,467 clusters for 54,575 documents, was evaluated against a YAGO category set of 73,944 categories and against a reduced set of 12,803 categories. Using the 73,944 categories for evaluation we achieved a micro purity of 0.3891 and a macro purity of 0.4945 in case of stability method as fitness measure. For 12,803 categories 0.4295 and

[1]http://www.inex.otago.ac.nz/tracks/wiki-mine/wiki-mine.asp
[2]http://inex.de-vries.id.au/scoreboard/

0.5303 were achieved. When using BIC as fitness measure we achieved 0.4959 and 0.5473 for 12,803 categories in the ground truth and 0.4493 and 0,4924 for 73,944 categories. The algorithm was executed on a quad core system with 16 GB RAM and required less than 4 minutes to compute the hierarchy (including vector loading), while taking advantage of all CPU cores (except for the top hierarchy level).

In case of the larger data set containing 2,666,190 documents our hierarchy consisted of 133,704 cluster nodes, yielding a micro purity of 0.4025 and a macro purity of 0.4457 compared to 348,552 YAGO categories. For the smaller set of 12,636 YAGO categories we achieved 0.4833 and 0.5359. Runtime was roughly about 2 hours on the same machine.

We have to admit that the evaluation of the INEX challenge was not intended to evaluate a hierarchical clustering solution. Nevertheless, the clusters appear to be reasonable pure considering the large-scale as well as multiple category distribution. Especially the clustering quality of higher levels in the topic hierarchy will disappear in this arithmetically averaged purity as there are much more leaf clusters than intermediate clusters. However, clusters nearer to the root will be of much more interest to the user since those will have the most influence on the guiding effect. For this reason the evaluation of our hierarchy should also contain a schema to assess the correctness of hierarchical relationships between documents and specifically weight the correctness or usability of higher levels in the hierarchy.

V. Conclusion & Future Work

Driven by the application scenario, in our approach we were focused on fast computation of the hierarchy, providing a browsing structure with reasonable labeling, and application of a projection algorithm to provide a visual exploration capability. Usability considerations as well as the scalability of the projection algorithms placed further constraints on the hierarchy structure which had to be addressed. Next development steps might include a dynamic selection of flat clustering algorithms, more sophisticated feature selection methods or clustering algorithms with implicit feature selection, such as the recently published ROCC algorithm [16]. Furthermore, we want to provide a distributed implementation of our approach to process Web-scale document collections. [3]

References

[1] R. Krishnapuram and K. Kummamuru, “Automatic taxonomy generation: Issues and possibilities.” in *IFSA*, vol. 2715. Springer-Verlag Heidelberg, 2003, pp. 52–63.

[2] D. R. Cutting, J. O. Pedersen, D. Karger, and J. W. Tukey, “Scatter/gather: A cluster-based approach to browsing large document collections.” in *SIGIR*. ACM Press, 1992, pp. 318–329.

[3] W. Ke, C. Sugimoto, and J. Mostafa, “Dynamicity vs. effectiveness: A user study of a clustering algorithm for scatter/gather.” in *SIGIR*. ACM Press, 2009, pp. 19–26.

[4] K. Andrews, W. Kienreich, V. Sabol, J. Becker, G. Droschl, F. Kappe, M. Granitzer, P. Auer, and K. Tochtermann, “The infosky visual explorer: exploiting hierarchical structure and document similarities.” *Information Visualization*, vol. 1, no. 3/4, pp. 166–181, 2002.

[5] V. Sabol, W. Kienreich, M. Muhr, W. Klieber, and M. Granitzer, “Visual knowledge discovery in dynamic enterprise text repositories.” in *IV*. IEEE Computer Society, 2009, pp. 361–368.

[6] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press, 2008.

[7] M. Daszykowski, B. Walczak, and D. L. Massart, “On the optimal partitioning of data with k-means, growing k-means, neural gas, and growing neural gas.” *Journal of Chemical Information and Computer Sciences*, vol. 42, no. 6, pp. 1378–1389, 2002.

[8] B. Fritzke, “A growing neural gas network learns topologies,” in *NIPS*. MIT Press, 1994, pp. 625–632.

[9] I. Dhillon, J. Fan, and Y. Guan, “Efficient clustering of very large document collections.” in *Data Mining for Scientific and Engineering Applications*, R. Grossman, C. Kamath, and R. Naburu, Eds. Kluwer Academic Publishers, 2001.

[10] Y. Zhao and G. Karypis, “Evaluation of hierarchical clustering algorithms for document datasets.” in *CIKM*, McLean, Virginia, 2002, pp. 515–524.

[11] S. Zhong, “Efficient online spherical k-means clustering,” *IJCNN*, vol. 5, pp. 3180–3185 vol. 5, July-4 Aug. 2005.

[12] Q. Zhao, M. Xu, and P. Frnti, “Knee point detection on bayesian information criterion.” in *ICTAI*. IEEE Computer Society, 2008, pp. 431–438.

[13] T. Lange, M. L. Braun, V. Roth, and J. M. Buhmann, “Stability-based model selection.” in *NIPS*. MIT Press, 2002, pp. 617–624.

[14] D. Carmel, H. Roitman, and N. Zwerdling, “Enhancing cluster labeling using wikipedia.” in *SIGIR*. ACM Press, 2009, pp. 139–146.

[15] D. Arthur and S. Vassilvitskii, “k-means++: the advantages of careful seeding.” in *SODA*, N. Bansal, K. Pruhs, and C. Stein, Eds. SIAM, 2007, pp. 1027–1035.

[16] M. Deodhar, G. Gupta, J. Ghosh, H. Cho, and I. Dhillon, “A scalable framework for discovering coherent co-clusters in noisy data.” in *ICML*. ACM New York, NY, USA, 2009, pp. 241–248.

[3] *Acknowledgment:* The Know-Center is funded within the Austrian COMET Program under the auspices of the Austrian Ministries BMVIT, BMWFJ and by the State of Styria, managed by the Austrian Research Promotion Agency FFG.