MODELING OF CONFLICTS FOR SCREEN MAP LABELING

I. Petzold [a,*], G. Gröger [b], L. Plümer [b]

[a] Dept. of Geography, University of Zürich, Winterthurerstrasse 109, CH-8057 Zürich, Switzerland - petzold@geo.unizh.ch
[b] Inst. for Cartography and Geoinformation, University of Bonn, Meckenheimer Allee 172, D-53115 Bonn, Germany - (groeger, pluemer)@ikg.uni-bonn.de

KEY WORDS: Cartography, GIS, Modeling, Data Structures, Real-time, Multiresolution, Generalization

ABSTRACT:

Map labeling is one of the most time-consuming and complex tasks during map generation. Additionally, if real-time map labeling is demanded like for screen maps, traditional automatic labeling approaches are not sufficient since they also do not support user interaction methods like zooming and scrolling – change of scale and map clip. Thus, we developed a new approach considering these specific demands for screen maps. The core of our approach is the modeling of possible label conflicts. This includes the identification of such conflicts depending upon their scale of occurrence and the storage of this information. Therefore, we designed a purpose-build data-structure, the so-called reactive conflict graph. This data-structure is built up before a user interacts with the system in the so-called preprocessing phase. It will be exploited during the interaction phase and ensures real-time labeling in a cartographic adequate way. The developed approach supports the labeling of point, line and area objects. The feasibility is successfully demonstrated in a component-based prototype.

1. INTRODUCTION

These days the use and importance of screen maps increases and they supersede more often traditional paper maps, especially in the context of mobile applications. The reason is their flexibility and adaptability to the users' specifications combined with the potential for interactive navigation in geographic information systems. This leads to specific demands on the functionality of screen maps, like scrolling and zooming.

Labeling is one of the most time consuming tasks in the map generation process, especially since each interaction leads to a completely new labeling and must be achieved in real-time. Scrolling demands not only the re-labeling at the clip border and the added map clip, but a complete new labeling. In addition, zooming a map not only requires changing the scale, but also a modification of the label size and is closely related to the (cartographic) generalization process.

The main goal of cartographic labeling is to avoid overlapping between labels, since this affects the readability of maps. Already for a fixed scale the determination of conflicts between labels is highly time-consuming and the major task during the labeling process. In general, there exist several possible labeling positions for each object, which are called "labeling space of the object". This requires the determination of potential conflicts between labeling spaces. These potential conflicts can be pre-computed and stored in a suitable data-structure to reduce the running-time during the main labeling process.

Due to the mentioned functionalities, screen maps are not restricted to a certain scale. Since the occurrence of potential conflicts depends on scale, the complexity of their determination and modeling rises significantly. Considering these difficulties, we developed a data-structure called "reactive conflict graph," which models the potential conflicts independent of scale. The conflict graph is a graph whose nodes are the objects to be labeled and whose edges represent the potential conflicts. To take the scale into account, the edges are attributed by the scale range of the occurrence of the potential conflict between the two objects, which are connected by the edge. So the reactive conflict graph enables efficient labeling.

Our concept of the reactive conflict graph covers point, line and area objects to be labeled and takes the different geometric shapes and properties into account.

During the zooming out, the number of conflicts increases. Thus, we have to omit objects to be labeled. Deselection criteria determine these objects, taking into account priorities and further cartographic constraints. These criteria ensure that the number of conflicts per object is reduced to an appropriate level.

2. CARTOGRAPHIC BACKGROUND

This section covers the necessary cartographic map labeling knowledge needed for understanding this paper as well as a short discussion about the difference between paper (static) and screen (dynamic) maps.

The obvious differences between paper and screen maps come from the media: The resolution and size of screen maps are a fraction of the resolution and size of paper maps. So the symbols in screen maps are less filigree and also the density of symbols for the same clip and scale is lower. Nevertheless, the labeling of screen maps should have the same high quality as paper maps.

Due to the media, screen maps can compensate the mentioned disadvantages thru interactivity. The interactivity results from the (screen map) functions of scrolling (changing the map clipping), zooming (free choice of scale) and integrating different thematic layers. As a result of the restrictions, the interactivity and the described functions, screen maps are short-lived. For a user-friendly interaction, these maps must be generated on the stroke-of-a-key. This time factor compensates

the disadvantage of labeling a reduced number of objects compared to paper maps.

For simplification, labeling algorithms like the proposed approach uses a rectangular approximation of the label.

It is obvious that the label size depends on the scale of the map. The size of the label is not growing or shrinking in proportion to the map scale, which causes the relabeling after each zooming – change of scale. The label size (*labSize*) could be approximated by an exponential function for a target scale (*scale*) in dependence on a reference size (*refLabSize*) and reference scale (*refScale*) as follows:

$$labSize = refLabSize\left(\frac{scale}{refScale}\right)^{v} \text{ with } v \in [0..0.15]$$ the growing and shrinking can be adjusted. More details can be found in previous papers (Petzold, 2003; Petzold et. al. 1999).

In contrast to other approaches we are using the concept of sliding labels or more precise *labeling spaces*. These labeling spaces represents the convex hull of all possible label positions and are simple computable geometric shapes like rectangular boxes for point objects.

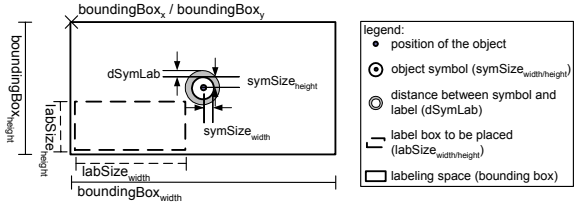


Figure 1. The labeling space of a point object

As shown in Figure 1 for point objects, the label positions are not restricted to a certain number, but the label can freely move or more precisely slide along the inside border of the labeling space. The overlapping with the symbol is not allowed.

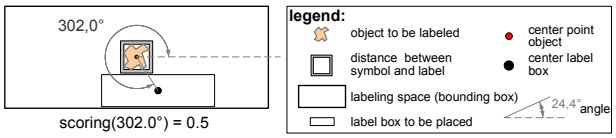


Figure 2. The continuous scoring of label positions in a labeling space – an example for a label position or rather angle and its scoring.

According to cartographic rules, some positions are in favor of others. This is covered by adapting the scoring of label positions as proposed in (Imhof, 1962).

For labeling point objects, a continuous scoring function using the angle between the horizontal line passing through the center of the point object and the center of the label position can be used (Figure 2).

The label of a line object wriggles along its symbol (Figure 3 a)). Due to the constant distance between the symbol of the object and the label, the concept of a buffer can be used to define the labeling space, as shown in Figure 3. This slight to huge buffer, especially at the end of the lines where a label will never be placed, will be taken into account for faster computation and easier handling. For scoring label positions, the distances of the label between start and end point of the line object, the twist of the line beneath or above the label will be used (Christensen, 1995; Edmondson, 1996; Petzold et. al., 1997).

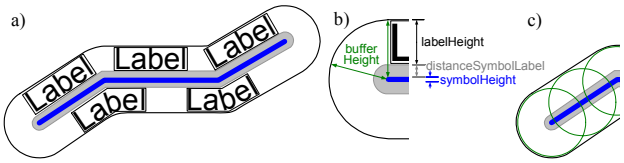


Figure 3. Labeling space of a line object is shown in a). It is derived by the buffer of the line object b) and c).

The labeling of area objects is reduced to the labeling of line objects. Otherwise the labeling space of an area object would correspond to its area and would increase the number of (potential) conflicts with other labels dramatically.

3. MODELING OF CONFLICTS

The modeling process is divided into two parts: First, the recognition of conflicts with accompanying algorithms, and second, the representation of them with accompanying data-structures. It is obvious that only this information is gathered which can be represented in the data-structure.

At first sight it seems paradox to model conflicts if the objective is to avoid conflicts, like the title of this paper pretends. However, it is less difficult to determine conflicts, to represent them in a data-structure, and finally to exploit this knowledge to obtain a conflict free labeling than the other way round as we will see in the following.

3.1 Characteristic of conflicts

This subsection deals with labeling conflicts and pictorially describes how they "emerge", "disappear" and which characteristics have conflicts in common.

In the following example, the focus is on two objects to be labeled and we will start the description in a huge scale where no labeling conflicts exist (Figure 4). If the scale gets smaller – zooming out – the objects move closer to each other, but the label size almost remains constant. The label shrinks much slower than the other map objects, as mentioned in the cartographic background section. At first the labels might move away from each other to avoid conflicts if there is enough space and the cartographic rules allow this.

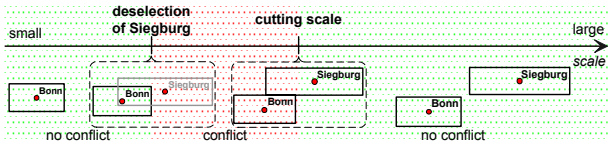


Figure 4. Life cycle of a conflict: 1) Before a conflict – large scale; 2) conflicts starts – cutting scale; 3) conflict ends / removal of one label – deselection scale.

Finally, this leads to a touch and later to an overlap between the labels of both objects. The scale of the first "touch" is called *cutting scale*. If we zoom out further, the overlapping area grows. This results in a very small scale where each label will overlap each other. It is obvious that it is not possible to label all objects. So we need a *deselection* criterion that decides below which scale an object will not be labeled any further. As we will see later this deselection scale belongs to the object and is "passed" to the conflict.

The first task is to develop a method to obtain the cutting scale. It would be too time consuming to determine the cutting scale with a nested-interval as described above. As mentioned in the description, the positions of the objects to be labeled and the label size changes, but both with different proportion relative to the scale. To achieve an analytic calculation of the cutting scale, the position of the objects are fixed to these ones of a specific scale (real world coordinates). Figuratively speaking, the objects remain at their original positions and instead the label size increases during decreasing the scale. To retrieve the appropriate scaled map, the coordinates have to be projected to screen map coordinates by a simple scale down. For simplification reasons, we assume that real world coordinates as well as the screen map coordinates are given in a Cartesian coordinate system. This approximation is of course only valid up to a certain scale, but suitable for screen maps.

The following subsection deals with the analytic calculation of the cutting scale in dependence on the involved object types.

3.2 Calculation of the cutting scale

3.2.1 Conflict between labeling spaces of two point objects

The cutting scale of the labeling space between two point objects has to be calculated separately for the horizontal and vertical direction.

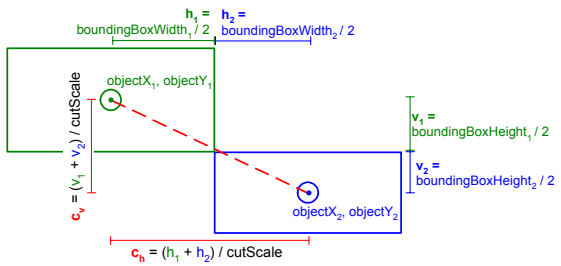


Figure 5 The cutting scale of point objects is calculated separately for the horizontal and vertical direction. c_h and c_v are the distances in the scale 1:1, so both addends must be divided by the cutting scale, the scale where the bounding boxes touches, to transform them to the scale 1:1.

We use the label size function presented in section 2 and describe the derivation of the cutting scale in the horizontal direction. The label size (*labSize*) for the cutting scale results from the distance between the two point objects (c) divided by the cutting scale (*cutScale*) as shown in Figure 5. The label size for the reference scale (*refLabSize*) will be replaced by the reference sizes of the half labeling space size, strictly speaking with the sum of the reference sizes of the half symbol size (*refSymSize*), the distance between symbol and label (*refDSymLab*) and the label size (*refLabSize*). After insertion of the parameters and resolving to the scale for the horizontal direction, the function is given by:

$$cutScale_h = \left(\frac{c_h}{\frac{refLabSize_{width,1} + refDSymLab_1 + \frac{refSymSize_{width,1}}{2}}{refScale_1^{\,v}} + \frac{refLabSize_{width,2} + refDSymLab_2 + \frac{refSymSize_{width,2}}{2}}{refScale_2^{\,v}}} \right)^{\frac{1}{v-1}}$$

The constant v in the exponent allows adjusting the growing and shrinking of the label.

The determination of the cutting scale for the vertical direction is analogous. The final cutting scale is the minimum of the cutting scales in both directions.

3.2.2 Conflict between labeling spaces of two line objects

The calculation of the cutting scale between the labeling spaces of two line objects is analogous to the previous one developed between point objects, except that there is no distinction between the horizontal and vertical direction. For simplification reasons, the line object consists of (simple) line segments. Well-known algorithms of computational geometry (Glassner, 1990) can be used to determine the shortest distance between both involved line objects in scale 1:1. The labeling space is identical to the buffer of the line object and thus its construction method can be exploited as shown in Figure 3. The calculation of the cutting scale can be expressed as follows:

$$cutScale = \left(\frac{d}{\frac{refLabHeight_1 + refDSymLab_1 + \frac{refSymHeight_1}{2}}{refScale_1^{\,v}} + \frac{refLabHeight_2 + refDSymLab_2 + \frac{refSymHeight_2}{2}}{refScale_2^{\,v}}} \right)^{\frac{1}{v-1}}$$

refLabHeight is the height of the label to the reference scale *refScale*, *refDSymLab* represents the distance between the symbol and the label, d is the shortest distance between both line objects in the scale 1:1 and with v in the exponent the growing and shrinking of the label can be adjusted.

3.2.3 Conflicts between labeling spaces of a point and a line object

The computation of the cutting scale between the labeling spaces of a point and a line object is more difficult because the extension of labeling space of point objects is different in horizontal and vertical direction. This leads to many special cases. The center of the point object and the four corners of its labeling space define four sectors (Figure 6). For each of these sectors, the shortest horizontal (for the sectors left and right of the center of the point object) or vertical (for the sectors above and below the center of the point object) distance between point and line object is determined. The calculations of the cutting scale considering all special cases can be found in (Petzold, 2003).

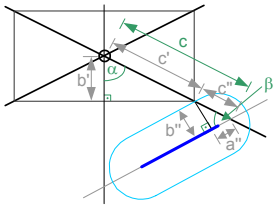


Figure 6 Determination of the cutting scale between point and line object to be labeled.

3.3 Deselection of labeling of objects

In this subsection we will look ahead to the final map labeling and the labeling algorithm. Under a certain scale the labeling spaces of all objects are in conflict with each other as mentioned in subsection 3.1 and visualized in Figure 7 a) and b). It is obvious that not all objects in Figure 7 a) can be labeled in the used scale. Labels have to been omitted – or in other words – *deselected*. A rough deselection can be performed before the labeling algorithm starts and leads to a reduced running time.

Therefore we need a criterion that decides for each object below which scale it is not further labeled (Figure 4). So this criterion belongs to the object to be labeled and not to the conflict, but it is passed to the conflict as described later in section 4.

We developed and examined a few criteria like node degree and conflict free space which are in detail discussed in (Petzold,

2003). A short overview can be found in (Petzold et. al., 2003). This paper is restricted to the criterion best suited for deselecting point and line objects: *conflict free space*. It is insignificant if the conflict partners are labeling spaces or labels of point or line objects or map objects, which should be considered as long as they can be represented by a bounding box.

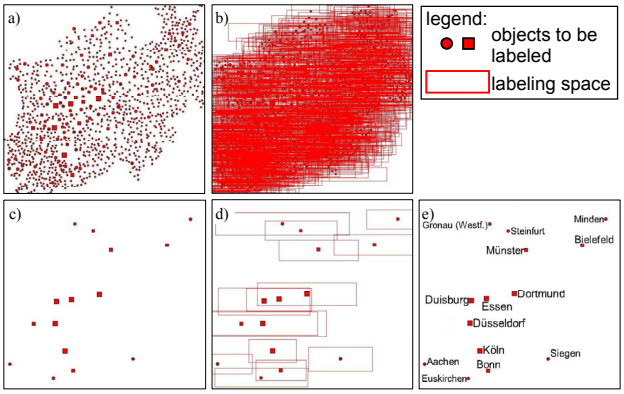


Figure 7. Impossibility to label all objects in small scales (a) and b)). Restriction to less objects necessary: deselection (c) to e)).

3.3.1 Deselection-criterion for point objects

For a specific scale, the difficulty to label a certain point object can be determined by the calculation of the remaining free label space. This can be done by considering the conflict labeling spaces or conflict labels of other objects.

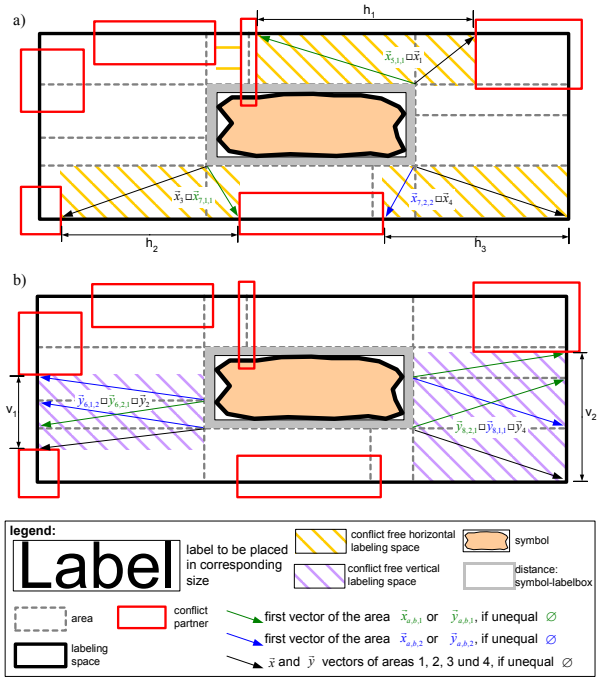


Figure 8 The criterion conflict free space is illustrated in both figures for the same example, separated for horizontal a) and vertical labeling spaces b).

Therefore the labeling space of a point object is split into areas (Figure 8). Each area has at most the size of the label. The distance between the symbol and the bounding box border is equal to the height in the vertical direction or rather the width in the horizontal direction of the label to be placed. So there is one row of these areas around the symbol (Figure 8).

For each area, the remaining conflict free labeling spaces are stored in vectors separately for the horizontal and vertical directions, as visualized in Figure 8. Conflict free labeling spaces for the horizontal sliding (above and below the symbol) guarantee sufficient conflict free space for the label height, but not automatically for the label width. This must be determined in a subsequent step described later. This procedure is analogous for the vertical sliding.

Each area in the corners of the bounding box has two vectors starting from the closest corner to the symbol – one for the horizontal and one for the vertical labeling spaces (Figure 8 a) and b)). Each area (directly) above and below the symbol has two vectors for representing the remaining horizontal space starting both from the closest corners to the symbol (Figure 8 a)). The areas (directly) left and right of the symbol are defined analogously for the vertical space (Figure 8 b)).

The conflict free labeling spaces represented by the vectors are evaluated separately for the horizontal and vertical sliding as shown in Figure 8. Thus the level of difficulty of labeling an object to a specific scale can be derived by the ratio free labeling space and the label size. A value greater one for one free space indicates that this space is huge enough to hold the label conflict free.

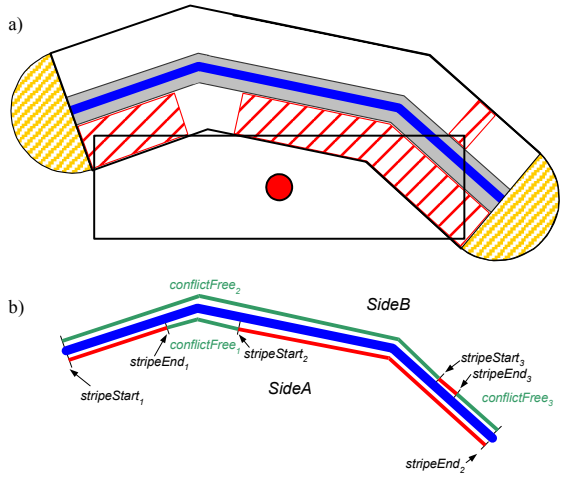


Figure 9 The criterion ‘conflict free space’ for line objects is a further development of the criterion for point objects. It can handle conflicts with line and point objects to be labeled a). The results are conflict free stripes b).

3.3.2 Deselection-criterion for line objects

For a line object, the remaining free label space can be calculated for a certain scale, too. In contrast to point labeling, the line labels wriggle along the line. Thus only the remaining free space in one direction – parallel to the line – has to be computed. This can be done by a “projection” of the conflict and conflict free spaces on the line object from both sides, as visualized in Figure 9.

4. DATA-STRUCTURE – REACTIVE CONFLICT GRAPH –

So far we have described methods for determining information about single conflicts. This section deals with assembling this information, the data-structure to store this information and how to build it up. As mentioned in the introduction to section 3, the developed data-structure will store the conflict information for

the labeling process. The structure is designed to allow an efficient labeling in a further step (section 5). In other words: it will react promptly to the users interaction during the interaction phase and delivers appropriate amount of detailed data on labeling requests. These kind of data-structures are called reactive data-structures (Oosterom, 1990) – in our case 'reactive conflict graph'.

For the data-structure we are using the well-known graph concept (Harary, 1972). A graph consists of nodes and edges; an edge connects two nodes. In our case each node represents one object to be labeled and each edge a potential conflict between both connected nodes or rather objects to be labeled. A potential conflict is an overlapping between two labeling spaces.

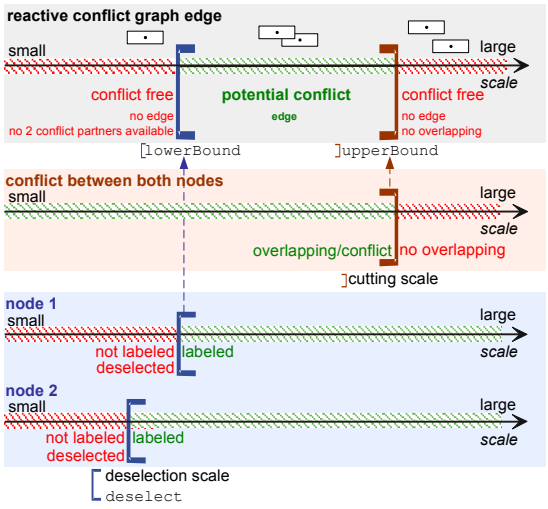


Figure 10 A conflict edge attributed by a scale interval (top axis) representing the conflict range. The lower bound is derived from the larger deselection scale of the involved nodes (node1 and node2 – two lower axes). The upper bound is the cutting scale between the labeling spaces of the involved nodes (second axis from top).

The characteristic of a conflict is discussed in subsection 3.1 in detail and visualized in Figure 4. It occurs in a scale-interval. The upper bound of the interval is the cutting scale and can be calculated as described in subsection 3.2.

The deselection scale for objects is determined by the deselection criterion that is developed for the different object types in subsection 3.3. This deselection-scale of each object is passed to the involved potential conflicts (Figure 10). A potential conflict disappears beneath the maximum of the deselection-scales of the both incident objects.

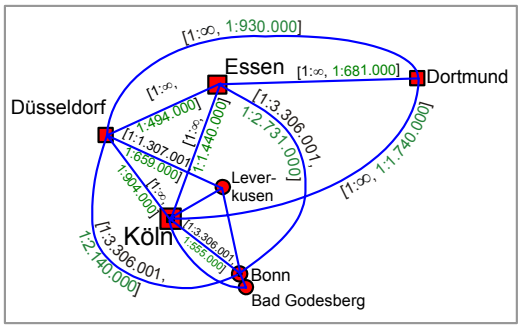
As you discovered, a procedure to generate the data-structure was not yet mentioned. In a naive approach, a potential conflict would be assumed between all pairs of objects and in a further step thinned out depending on the scale-interval of the conflict edges. This would lead to a high running-time because a lot of unnecessary conflicts would be assumed and afterwards deleted.

To avoid these disadvantages we developed an algorithm to speed up the generation of the data-structure (Algorithm 1). The data-structure is built up depending upon the object's priority. It starts with the least important object and considers possible conflicts to objects with higher priorities. *MaxDifficult* represents the lower bound of the number of allowed possible conflicts. In combination with the previously developed deselection criterion, this is the minimum remaining conflict free space. The allowed conflicts cannot be chosen only by their priority, the scale and distance of occurrence have to be considered in addition. This is performed in the inner loop through the sorted cutting-scale-list.

Input: objects to label with position, reference labeling, priority and *maxDifficulty* (maximum labeling difficulty relatively to more important objects)
Output: reactive conflict graph (nodes and edges)

- **for** each object in ascending order of priority (*priorityObject*):
- generate node
- calculate cutting-scale of *priorityObject* to all more important objects
- sort these objects in descending order of cutting-scale
- **run through** cutting-scale-list (*cutObject*):
- calculate labeling difficulty between *priorityObject* and *cutObject* and all previous elements of cutting-scale-list
- **if** calculated labeling difficulty is higher than *maxDifficulty*:
- **then** generate edge between *priorityObject* and *cutObject*
- set *upperBound* to cutting-scale
- **else** leave loop
- **if** *cutObject* exists
- **then** set *deselect* on cutting-scale of *cutObject* + ε
- **else** set *deselect* on 0
- **for** each pair of objects with same priority:
- calculate cutting-scale
- generate edge if cutting-scale greater than both *deselect* of incident nodes and set *upperBound* to calculated cutting-scale
- **for** each edge:
- set *lowerBound* to maximum of *deselect* of incident nodes
- eliminate edge if *upperBound* smaller than *lowerBound*

Algorithm 1 Generation of the reactive conflict graph.



Bonn - Leverkusen	[1:1.307.001, 1:1.262.000]
Bonn - Bad Godesberg	[1:2.247.001, 1:100.000]
Köln - Leverkusen	[1:1.307.001, 1:274.000]
Köln - Bad Godesberg	[1:2.247.001, 1:681.000]

Figure 11 Visualized reactive conflict graph with node labels and edges attributed with scale intervals. Due to space constraints, some of the scale intervals have to be listed below the figure.

5. LABELING

For fast labeling and access, the reactive conflict graph is stored in a three dimensional geometric data-structure like the 3-D-R-Tree (Guttman, 1984). The first two dimensions represent the geometry while the third dimension represents the scale. The objects' geometry or its bounding box is stored. For the labeling, the reactive conflict graph is queried to a specific map clipping and scale. The result – the so-called static conflict graph – contains all objects to be labeled and all possible conflicts between these objects for the specific map clipping and scale. Details about the usage and labeling algorithms are described in detail in (Petzold, 2003) and in short in (Petzold et. al., 2003).

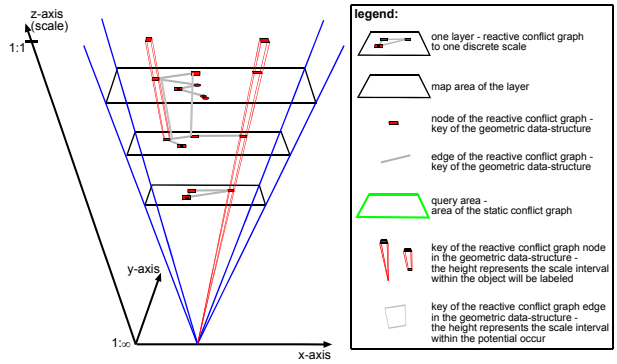


Figure 12 Embedding of the reactive conflict graph in a 3-dimensional data-structure and the derivation of the static conflict graph.

6. ANALOGY TO GENERALIZATION

The operation zooming for map labeling encloses aspects of cartographic generalization for the labeling, mainly the selection of objects to be labeled or the other way around, the deselection of objects. For the deselection we developed a few cartographic criteria and discussed the conflict free space criterion for point and line objects in subsection 3.3. This kind of criterion can also be applied to cartographic generalization. In addition, a simplified case of displacement is treated by the shifting of label positions.

The usage of an applied reactive conflict graph for the generalization process will be also a benefit especially for multiresolution aspects. In this data-structure, all information necessary for the generalization process can be stored and it enables an easy and quick access to these information. Comparable to the introduced labeling process, generalization algorithms can be based on this data-structure.

7. CONCLUSIONS

In this paper we presented an approach for modeling conflicts for efficient label placement. It supports the labeling of point, line and area objects and the operations of zooming and scrolling. These operations are essential for screen maps. Together with the developed two-phase-approach it enables labeling in real-time, which means in a stroke-of-a-key. Cartographic priority of positions, and minimizing of information-loss are considered as well.

In contrast to our previous paper (Petzold et.al., 2003) we put the focus here on the modeling of labeling conflicts. This encapsulates which label information should be collected, how it can be calculated and how to represent this information in a purpose-build data-structure, the reactive conflict graph. The generation of the reactive conflict graph is done in the so-called preprocessing-phase.

With the reactive conflict graph, the running-time for the final labeling in the interaction-phase can be reduced by shifting time-consuming calculations in a so-called preprocessing-phase as far as possible. Additionally, heuristics are used to reduce the running time in both phases.

Essential for our concept is the determination of the labeling difficulty, which is used to make a preselection of the objects to be labeled, depending on object priority and scale. In addition, this concept relaxes the restriction to discrete label positions which is employed in many other approaches and satisfies cartographic principles.

There is a strong analogy to cartographic generalization, especially regarding the selection/deselection process and its dependence on scale and potential to resolve conflicts, as well as how conflicts are modeled. Thus the concept presented in this paper can be extended to cartographic generalization.

The feasibility of the concept and the labeling in real-time is demonstrated by a Java prototype. The point labeling is implemented completely, while the line and area labeling is realized only partially.

References

Christensen, J., 1995. Managing Designs Complexity: Using Stochastic Optimization in the Production of Computer Graphics. Ph.D.-Thesis, Harvard University, USA.

Edmondson, S., Christensen, J., Marks, J., Shieber, S., 1996. A General Cartographic Labeling Algorithm. Manuscript, Harvard University, Cambridge, Massachusetts, USA.

Glassner, A., 1990. *Graphics Gems*. Academic Press, San Diego, USA.

Guttman, A., 1984. R-trees: A Dynamic Index Structure for Spatial Searching. In: *Proc. of ACM SIGMOD Conference on Management of Data*, University of California, Berkeley, USA, pp 47-57.

Harary, F., 1972. *Graph Theory*. Addison Wesley.

Imhof, E., 1962. Die Anordnung der Namen in der Karte. In: *International Yearbook of Cartography II*, Kirschbaumverlag, Bonn.

Oosterom, P., 1990. Reactive Data-Structures for Geographic Information Systems. Ph.D.-Thesis, Department of Computer Science, Leiden University, Netherlands.

Petzold, I., Plümer, L., 1997. Plazierung der Beschriftung in dynamisch erzeugten Bildschirmkarten. In: *Nachrichten aus dem Vermessungswesen*, Reihe I, Heft Nr. 117, Verlag des Instituts für Angewandte Geodäsie, Frankfurt a. M..

Petzold, I., Plümer, L., Heber, M., 1999. Label Placement for dynamically generated screen maps. In: *Proceedings of the Ottawa ICC99*, Canada.

Petzold, I., 2003. Beschriftung von Bildschirmkarten in Echtzeit – Konzept und Struktur. Ph.D.-Thesis, Institute of Cartography and Geoinformation, University of Bonn, Bonn. http://hss.ulb.uni-bonn.de/ulb_bonn/diss_online/landw_fak/2003/petzold_ingo/index.htm

Acknowledgements

The research and the prototype were funded by the DFG (German Research Foundation) in the project “Labeling Screen maps in real time”. We thank Dirk Burghardt and Robert Weibel (both Geographic Information Systems Division, Department of Geography, University of Zürich, Switzerland) as well as Axes-Systems AG in Alpnach, Switzerland for their support, which enabled us to write the paper.