

Student thesis series INES nr 521

# Automatic label placement for city maps with the labelling library PAL

**Pontus Cederholm**

---

2020  
Department of  
Physical Geography and Ecosystem Science  
Lund University  
Sölvegatan 12  
S-223 62 Lund  
Sweden



Pontus Cederholm (2020).

*Automatic label placement for city maps with the labelling library PAL*

*Automatisk text- och symbolsättning för stadskartor med algoritmprogrammet PAL*

Master degree thesis, 30 credits in *Geomatics*

Department of Physical Geography and Ecosystem Science, Lund University

Level: Master of Science (MSc)

Course duration: *January 2020 until June 2020*

#### Disclaimer

This document describes work undertaken as part of a program of study at the University of Lund. All views and opinions expressed herein remain the sole responsibility of the author, and do not necessarily represent those of the institute.

# Automatic label placement for city maps with the labelling library PAL

---

Pontus Cederholm

Master thesis, 30 credits, in *Geomatics*

Supervisor:

Lars Harrie

Department of Physical Geography and Ecosystem Science,  
Lund University

Exam committee:

Per-Ola Olsson and Helen Eriksson

Department of Physical Geography and Ecosystem Science,  
Lund University



## Acknowledgements

Firstly, I would like to thank my supervisor Lars Harrie for giving invaluable advice and support throughout this thesis project. I also want to thank the staff at T-Kartor, for making this interesting project possible, for taking their time in providing the information and data needed, and for letting us visit them in Kristianstad. I acknowledge the QGIS community, which has been an important source for everything concerning QGIS.

## Abstract

In cartography and map making, the placement of labels such as texts is an art that has been practiced manually for a long time. Labels describe what their corresponding map features represent. In recent decades, techniques for automating map label placement have been developed. There are different types of algorithms that can be applied to solve the label placement problem, including combinatorial optimization methods, sequential methods, and slider methods. In this study, an interview was conducted to retrieve label placement rules applied by the company T-Kartor in their production of City Wayfinding maps. The labelling library PAL was used in the open-source GIS application QGIS. PAL parameters were set in a label placement program developed, which produced map labelling meant to follow the label placement rules. The results showed that the quality of the automated label placement produced by the program was too poor for professional usage, though the combinatorial optimization method used in PAL could potentially produce satisfactory labelling if utilized in a better way. This study could form the basis for further research on the topic. Future studies should test and compare optimization methods or algorithm types, since this can affect the label placement quality significantly.

## Sammanfattning

Inom kartografi och kartframställning har textsättning länge gjorts manuellt. Karttexter förklarar vad deras kartobjekt representerar, och under de senaste decennierna har metoder för att skapa automatiserad textsättning utvecklats. Det finns olika typer av algoritmer som kan användas för att hitta textsättningslösningar, till exempel kombinatoriska optimeringsmetoder, sekventiella metoder, och slider-metoder. I den här studien har en intervju gjorts för att inhämta text- och symbolplaceringsregler som företaget T-Kartor tillämpar i deras produktion av City Wayfinding-kartor. Textsättningsmjukvaran PAL har använts i GIS-programmet QGIS. PAL-parametrar har satts i ett text- och symbolplaceringsprogram så att placeringsreglerna ska följas. Kartresultaten producerade av programmet visade att kvalitén på den automatiserade text- och symbolsättningen inte var bra nog för att användas i professionellt syfte, men de kombinatoriska optimeringsmetoder som PAL tillämpar skulle potentiellt kunna producera god text- och symbolplacering om de används på ett bättre sätt. Den här uppsatsen kan ligga som grund för vidare studier på ämnet. Framtida forskning bör

testa och jämföra optimeringsmetoder eller algoritmtyper, eftersom det kan påverka text- och symbolsättningskvalitén på ett betydande sätt.

# TABLE OF CONTENTS

<b>1. INTRODUCTION</b> .....	<b>1</b>
1.1 BACKGROUND .....	1
1.2 PROBLEM STATEMENT.....	2
1.3 AIM .....	3
1.4 STUDY DESIGN.....	3
1.5 DISPOSITION .....	3
<b>2. LABELLING RULES</b> .....	<b>4</b>
2.1 LABEL POSITION .....	4
2.1.1 <i>Point feature labelling</i> .....	4
2.1.2 <i>Line feature labelling</i> .....	5
2.1.3 <i>Polygon feature labelling</i> .....	6
2.2 OTHER LABELLING CONSIDERATIONS.....	7
<b>3. METHODS FOR AUTOMATED LABELLING</b> .....	<b>8</b>
3.1 COMBINATORIAL OPTIMIZATION .....	8
3.2 SEQUENTIAL PLACEMENT .....	12
3.3 SLIDER MODEL.....	15
<b>4. LABEL PLACEMENT METHOD AND TOOLS FOR PRACTICAL STUDY</b> .....	<b>16</b>
4.1 SELECTION OF METHOD AND TOOLS .....	16
4.2 THE LABELLING LIBRARY PAL .....	17
4.2.1 <i>PAL algorithms</i> .....	17
4.2.2 <i>PAL implementation in the QGIS user interface</i> .....	19
4.2.3 <i>QGIS Python API for labelling</i> .....	23
4.2.4 <i>PAL applied in research</i> .....	24
<b>5. PRACTICAL STUDY OF LABEL PLACEMENT</b> .....	<b>25</b>
5.1 LABELLING RULES CURRENTLY APPLIED BY T-KARTOR .....	25
5.1.1 <i>Point feature labelling</i> .....	26
5.1.2 <i>Line feature labelling</i> .....	27
5.1.3 <i>Polygon feature labelling</i> .....	27
5.1.4 <i>Symbols and icons</i> .....	28
5.1.5 <i>Label overlapping and removal</i> .....	30
5.1.6 <i>Additional rules and considerations</i> .....	30
5.1.7 <i>Comparison with the labelling rules from the literature</i> .....	31
5.1.8 <i>Labelling rules applied in this study</i> .....	32
5.2 WORKFLOW .....	32
5.3 DATA .....	34
5.4 LABEL PLACEMENT PROGRAM .....	35
<b>6. RESULTS</b> .....	<b>38</b>
6.1 EVALUATION ACCORDING TO T-KARTOR'S LABELLING RULES.....	38
<b>7. DISCUSSION</b> .....	<b>43</b>
7.1 INTERPRETATION OF RESULTS.....	43
7.2 ALGORITHM SUITABILITY.....	45
7.3 AUTOMATED AND MANUAL LABEL PLACEMENT .....	47
7.4 FUTURE RESEARCH .....	47
<b>8. CONCLUSIONS</b> .....	<b>48</b>
<b>REFERENCES</b> .....	<b>49</b>
<b>APPENDIX A: PROGRAM CODE</b> .....	<b>51</b>







# 1. Introduction

## 1.1 Background

A map can be defined as something that visually illustrates shapes and relationships in the spatial domain (Robinson et al. 1984). There are two main map types. General-reference maps are designed to inform on where real-world features are located, e.g. the location of mountains, lakes, and buildings. Topographic maps are a common type of general-reference maps. Thematic maps on the other hand, focus on one or several attributes (themes), and the patterns in the two-dimensional plane this data convey, e.g. patterns of life expectancy or annual precipitation (Slocum et al. 2005). The maps discussed in this study are general-reference maps.

Cartography essentially involves studying, designing, and producing maps (Robinson et al. 1984). Map making has been an object of study for many centuries, and it is still to this day an active research field, in which new technologies are applied continuously. The rapid development in computer technology and computer science during the past decades has been significant for the field of cartography since more and more aspects of the production of maps could be automated. It also created new possibilities of performing different kinds of spatial analysis (Kraak and Ormeling 1996; Longley et al. 2011).

For map users to better understand what the different map features represent, texts, or labels, are needed on maps. Without them maps would be much harder to interpret. The text referred to does not include supplementary text that comes with a map, e.g. the legend text, the title, or the text for giving credits, but only the text that is inside the map frame (Kraak and Ormeling 1996). It is important that the text labels are legible, since they are meant to be read. A text also needs to be easily linked to its feature (Slocum et al. 2005). When a label is attached to a point, line, or polygon map feature, it is desired that it is placed in a way that supports the geographic shape and extent of the map feature (Robinson et al. 1984). For example, a line feature text should be placed parallel with the line it belongs to. Here, a minor note must be mentioned: *Lettering* refers to everything that is related to adding texts to maps (Robinson et al. 1984). To avoid confusion, and to also include symbols and icons, the term *labelling* is used henceforth. Similarly, the term *label* is used as a collective term to include texts, symbols, and icons.

There are several fundamental rules one should have in mind when labelling a map. For example, overlapping or overprinting (i.e. a feature or text is unintentionally covered by

another text) during labelling should be minimized. These rules, and many more, are described and discussed thoroughly by e.g. Robinson et al. (1984) and Slocum et al. (2005) and reviewed in section 2.1. Apart from text placement, there are a few other considerations to be made when labelling. Those deal with the appearance of the text that is not related to placement, e.g. the style, size, and color. This is discussed in section 2.2. However, this thesis mainly focuses on the placement aspect of labelling.

## 1.2 Problem statement

In the process of map production, manual placement of labels can be one of the most time-consuming tasks (Yoeli 1972), and it can also be very costly (Robinson et al. 1984). This issue has led to the development of algorithms used to automate label placement. The issue of labelling is quite easy to understand intuitively, but the problem can become complex very fast with increasing amounts of input data. Maps often have a large number of features to be labelled, and those labels can be placed at many different locations close to their corresponding features. This results in many possible label placement combinations. Due to the complexity that arises from this, algorithms that apply heuristic methods have been created so that the labelling can be practically solved, with solutions that are not optimal but often relatively good.

Such algorithms can be used by different actors that are working within the field of cartography and map production. T-Kartor is a company that produces, among other things, City Wayfinding maps for many cities across the world. These maps are static, large scale, and show the local area. They are set up to inform pedestrians of the city on how to navigate in the area. T-Kartor use ESRI's labeling extension Maplex together with ArcGIS to generate automatic labelling. They estimate that the Maplex program has a label placement accuracy of approximately 70%, with the remaining labels edited manually in Adobe Illustrator in the post-processing of the map. However, the company assesses the possibility to move their map production line from ArcGIS to the open-source application QGIS (QGIS Development Team 2020b), which they find focuses more on developments in the field of cartography. Therefore, they need a new program to replace the Maplex Label Engine.

### 1.3 Aim

The general aim is to minimize the manual workload on label placement and cartographic design, by implementing automated label placement. The specific aim of this project is to design and evaluate a label placement method that produces good labelling for static, large scale city maps. The labelling should follow rules established by the cities and by the map company T-Kartor. A specific objective is to find a new method of using open-source GIS and programming software to develop an easily accessed label placement program.

### 1.4 Study design

This thesis is composed of a theoretical study and a practical study. In the theoretical part, a literature study on labelling rules is presented, where label placement for point, line, and polygon features are introduced. An interview was conducted, where a person at T-Kartor was asked questions regarding the labelling rules implemented by the company. The information compiled from this interview forms the basis of the labelling rules applied in the practical study. After this, methods of automated labelling were searched for in the literature, which was used to form the choice of labelling algorithm. In the practical study, the theoretical knowledge and rules are meant to be applied and followed in the label placement program that is developed. The labelling results are discussed, as to whether it succeeded to meet the labelling requirements or not, and if another automated labelling method could better follow the labelling rules.

### 1.5 Disposition

This thesis consists of eight sections. In the first section, the background of the study is covered, the problem statement is defined, and the aim is stated. The general design of the study is also explained. In the second section, labelling rules found in the literature is reviewed. Algorithm methods for automated label placement are reviewed in section three. The choice of method for the practical study is explained and described in section four. In the fourth section the algorithm method is also reviewed.

Section five details the practical part of the study, reviewing the label placement rules applied, describing the workflow, the data used, the label placement program developed, and the method of evaluation. The results of the practical study are found in section six. Here the labelling is evaluated according to T-Kartor's label placement rules. In the discussion, the

seventh part of the thesis, the results are interpreted. This is followed by a discussion concerning algorithm suitability, and whether other methods could achieve better label placement quality than the one used. Automated and manual label placement is also discussed. The last part of the discussion concerns what future research could focus on. Lastly, section eight concludes the thesis.

## 2. Labelling rules

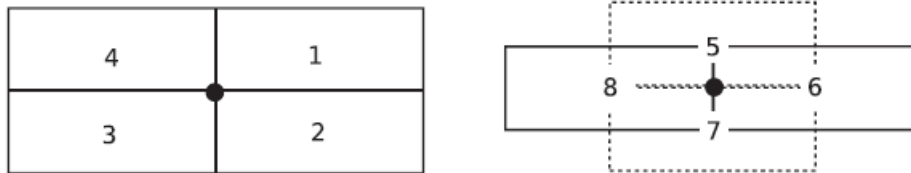
### 2.1 Label position

Rules for label placement promote legibility and feature recognition, but they are also important from an aesthetic perspective (Imhof 1975). Some labelling rules can be applied to point, line, and polygon features; these are reviewed first. A label should be placed in a way that effectively informs on the location of the feature (Robinson et al. 1984), and so that there is no question of which feature the label belongs to (Imhof 1975; Slocum et al. 2005). Another principle is that labels should not be printed upside down, since this would not make them legible. This means that labels are always put in a left-to-right direction (Robinson et al. 1984; Slocum et al. 2005). The letter spacing of texts should be minimized, especially for point feature texts. Different letter and word spacing should mainly be applied for different polygon features, since the spacing often is used to suit the shape and spread of a polygon (Slocum et al. 2005). Text should not cross or cover coastlines or other land-water boundaries. Moreover, straight, horizontal labels are generally preferred over curved labels. However, this is not true for labelling line and polygon features that are not horizontal: here the label should follow the feature direction (Slocum et al. 2005). Overlaps, i.e. when a label and a map feature are placed on the same location, should be prevented if possible. In overlap situations, the labels should be left unchanged and the features should be edited, e.g. by using a mask to cover the feature (Robinson et al. 1984; Slocum et al. 2005).

#### 2.1.1 Point feature labelling

Since labels describing point features cannot be on the feature in the same way that labels for line or polygon features can be, they need to be placed somewhere around the point. The most preferred label position around a point feature varies, but it is common to have it placed above and to the right of the point (Imhof 1975). Straight above or below the point is also common (Robinson et al. 1984). However, the prioritization can vary quite much, e.g. see Jones (1989).

An example of how label candidates can be prioritized is given by *figure 2.1*. If a placement would result in overlaying of a feature, then another position should be chosen. This can be done with having a defined prioritization of preferred placements. Having the label just to the left or to the right of the point, i.e. aligning the point and label on a single line, is not preferred. This is because it can decrease the legibility (Slocum et al. 2005).



*Figure 2.1: Eight different label position candidates for a point feature. The lower the number, the higher the prioritization. Modified after Marín and Pelegrín (2018).*

A point and its label should not be separated by another feature (Robinson et al. 1984; Slocum et al. 2005). If a point is on a coastline, the label should be inside the land area or inside the water area, and not on both. Where several lines of text are needed, the type of line alignment should be depending on how they are placed in relation to the feature: centered if below or above point; right-aligned if to the left of point, and vice versa (Slocum et al. 2005).

### 2.1.2 Line feature labelling

Line feature labels should be placed along the line features they describe, with no other feature lying between them. The text is better placed at parts where there is little other map information. The text should generally be curved so that it corresponds to the curve of the line (Robinson et al. 1984), but the text should not be excessively curved (Imhof 1975). This is illustrated in *figure 2.2*, where a straight text following a straight portion of the line feature is preferred over a very curved text. It is preferred that line feature text is placed above rather than below a line (Robinson et al. 1984; Slocum et al. 2005). For longer lines, the label can rightly appear several times, as opposed to increasing the spacing of the text. As stated above, labels should always have a left-right direction. In situations where a vertical line is to be labelled, the label should be read in a down-up direction (Slocum et al. 2005).

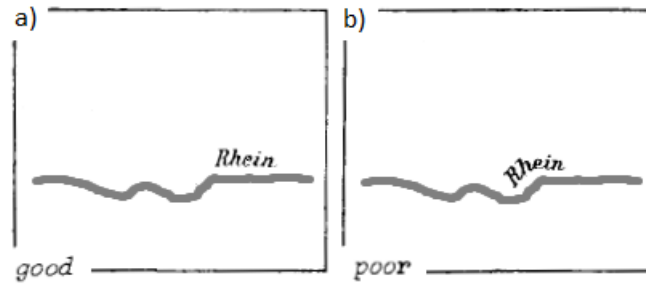


Figure 2.2: A line feature with a text label placed in two different ways: a) Good label placement b) Poor label placement. Modified after Imhof (1975).

### 2.1.3 Polygon feature labelling

The most important rule regarding labelling polygon features is that the label should be placed inside the polygon, if no other type of placement method is explicitly desired, or if the label cannot fit within the area (Imhof 1975). In those cases, it can be placed entirely outside of the feature (Robinson et al. 1984; Slocum et al. 2005). Furthermore, labels should not be too close to the polygon boundary (Robinson et al. 1984). The label should be placed in the center of the polygon part shown, and the label must not cover an excessive amount of the area (Slocum et al. 2005). The label can be curved but not excessively so. The letter and word spacing, the line spacing, the curvature, and the placement inside a polygon should match the shape of the polygon. However, the letters and lines should not be spaced too wide (Robinson et al. 1984; Slocum et al. 2005). Larger rivers and other line-like objects can be labelled as polygons when needed (Slocum et al. 2005). An example of polygon feature labelling is shown in *figure 2.3*.

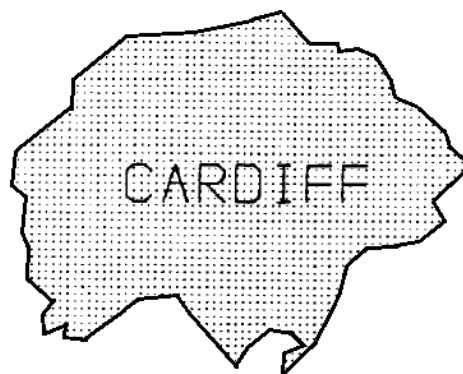


Figure 2.3: A polygon feature with its label placed inside it. Modified after Cook and Jones (1990).



## 2.2 Other labelling considerations

One important aspect is how to combine the placement of point, line, and polygon labels. In *figure 2.4*, labels have been added to point, line, and polygon features, though most of them are point feature labels. It is a good example of how point feature labels can have different placements to avoid overlapping.



*Figure 2.4: An example of point, line, and polygon feature labelling results. Modified after Edmondson et al. (1997).*

In addition to the positioning of labels, the appearance of them is also important. Here the style, form, size, and color are discussed. Style is the serifs used and other design aspects, e.g. the letter thickness applied. The letter form sans-serif is a common type used in cartography, due to its simple and clear appearance (Robinson et al. 1984). When using different styles in the labelling, they must easily be distinguished from one another, e.g. by having different sizes or thicknesses. This is often done to create label hierarchies (Kraak and Ormeling 1996).

There are two different forms for text: uppercase and lowercase. Often, labels that are more prioritized are in uppercase (Robinson et al. 1984). Uppercase letters can be used to label polygons (Slocum et al. 2005). These are better suited for increasing the letter and word spacing than lowercase letters. Slocum et al. (2005) state that letters should not be spaced more than four times the size of the letters.

The text size, i.e. the height of the text, should not be too small, since that can decrease the legibility. The minimum text size that is readable depends on the reading distance. Different label classes can have different sizes, and the size difference between those should ideally be 25 % or more for map users to easily distinguish classes (Robinson et al. 1984). In *figure 2.4* the polygon label *Vopika* has a distinctly larger text size than the other labels, which makes it easy to interpret it as an area label.

Lastly, the color used when labelling is of importance. There needs to be a color contrast between the labels and the background for the labels to be readable. This can be achieved by

having the background in a lighter color and the labels in a darker color (as on this page), or vice versa (Robinson et al. 1984). Coloring labels differently can be used to emphasize different categorical groups, e.g. countries, lakes, mountain ranges. This make such map features easier to find (Kraak and Ormeling 1996).

### 3. Methods for automated labelling

Several types of methods for automated label placement have been created and studied. This study reviews three main methods: combinatorial optimization, sequential method, and the slider model.

#### 3.1 Combinatorial optimization

One heuristic method that has been developed is an optimization technique, which stems from the field of combinatorial optimization. This method aims at optimizing the label placement by selecting an optimal label placement combination, or solution, based on an objective function that analytically describes the quality of the labelling solution. To find the optimal labelling solution, an optimization algorithm is used. The combinatorial optimization problem of map label placement is essentially NP-hard, signifying that no polynomial time-algorithm solving the problem perfectly has been developed yet (and there is doubt that such exist) (Christensen et al. 1995). This is the reason why such heuristic methods are applied.

An objective function quantifies the label placement quality of a labelling solution (Christensen et al. 1995). Thus, each solution has its own objective function value. An objective function is formed by summing factors that are weighed. The total cost of the summed factors with weights defines the objective function value of a label placement solution, with lower value signifying higher label placement quality. Therefore, the labelling solution with the lowest objective function value is the most preferred solution (Christensen et al. 1995). The construction of the objective function is important since it directly affects which solution that is finally selected by the algorithm.

Christensen et al. (1995) based their objective function definition on the work of Yoeli (1972), where the following factors are included: (1) The number of overlaps the label placement cause; (2) Candidate label placement prioritizations; (3) The number of deleted labels. Edmondson et al. (1997) applied a combinatorial optimization method for point, line, and polygon feature labelling. First, candidate positions were generated, and labels were

selected for each feature, creating a first label placement solution. Then, a labelling evaluation was performed, where (1) label placement (positioning metrics) and (2) resulting overlaps (overlap metrics) were taken into consideration to form an objective function. The objective function weighs the sum of point, line, and area-positioning metrics applied on all placed labels. The point-positioning metrics for different candidate positions are found in *figure 3.1*. The line-positioning metrics are more complex, where label flatness and label-line distance are considered, among other factors. The area labelling metric says that labels should be as close to the area centroid as possible. The weight values applied in the study are presented in *figure 3.2*. It shows how overlaps are penalized, and the placement criteria weights for the feature types.

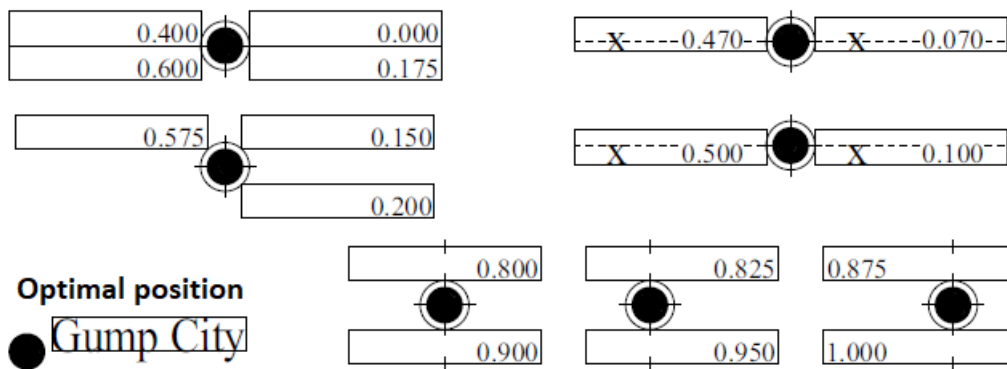


Figure 3.1: A point feature's candidate positions for its label. The candidate position metrics reflect how preferred the positions are, with lower values indicating better label placement. Modified after Edmondson et al. (1997).

Overlap metrics	
<i>PointOver</i>	10
<i>LineOver</i>	15
<i>AreaOver</i>	10
<i>LabelOver</i>	40
Positioning metrics	
<b>Point positioning</b>	
<i>PointPos</i>	1
<b>Line positioning</b>	
<i>AveDist</i>	1
<i>Flatness</i>	1
<i>MinDist</i>	NA
<i>Centeredness</i>	3
<i>Aboveness</i>	0.25
<b>Area positioning</b>	
<i>AreaPos</i>	10

Figure 3.2: The metric weights Edmondson et al. (1997) applied in the objective function. Modified after Edmondson et al. (1997).

Zoraster (1997) applied another objective function, based on four factors: (1) The distance from the placed label to its most optimal position; (2) Label deletion; (3) Label-label overlaps; (4) Label-symbol overlaps. The distance factor has the lowest cost, and the overlap factors have the highest costs. This means that overlaps are strongly discouraged, and that labels can be placed at non-optimal positions without necessarily producing a high objective function value.

Zhang and Harrie (2006a) developed a label placement method for real-time maps. Their objective function ( $f$ ) is defined by *equation 1*.

$$f(\text{label positions}) = w_1 * \Sigma \text{ placement properties} + w_2 * \Sigma \text{ cartographic disturbance} + w_3 * \Sigma \text{ label overlap}, \quad \text{Equation 1}$$

where  $w$  represent the weights of three (1, 2, 3) factors: (1) *Placement properties*, which determines how good the label-feature relationship is; (2) *Cartographic disturbance* quantifies how much the labels cover and thereby disturb other map information; (3) *Label overlap* is the overlapping proportion of a label exposed to overlap (where 0.05 is the highest proportion permitted).

These studies have a few factors in common in their way of constructing an objective function. They all include a placement factor, although they are designed differently. Some apply candidate label placement prioritization (e.g. Edmondson et al. 1997) and others compute the distance from the best candidate position (Zoraster 1997). Another shared factor concern overlapping. There are several types of overlaps defined in the studies, e.g. label-label, label-symbol, and labels obscuring other map objects. Some use the number of overlaps (Christensen et al. 1995; Edmondson et al. 1997), while others compute the overlapping ratio of labels (Zhang and Harrie 2006a).

The factor weights determine the relative importance of each factor. Edmondson et al. (1997) determined the weights values through intuition, which may not be the optimal method. Zhang and Harrie (2006a) assigned the highest weight value to the *label overlap* factor ( $w_3$ ), and the lowest weight value to *placement properties* ( $w_1$ ). Edmondson et al. (1997) also assigned the overlap factors higher weights. By giving an overlap factor a relatively high weight, overlaps may be minimized in a label placement solution.

To find an optimal label placement solution with a low objective function value, an optimization algorithm is implemented. Christensen et al. (1995) and Edmondson et al. (1997)

applied an optimization method called simulated annealing (Russell and Norvig 1995), which iteratively improves the labelling solution. The algorithm functions as follows:

From the features' candidate positions, label positions are randomly selected for every feature, creating a random labelling solution. A starting value for a temperature value  $T$  is set; the temperature controls the likelihood of escaping a local minima during the execution of the algorithm. The temperature is lowered by 10% in each iteration, i.e., the likelihood of escaping a minima is decreasing during the execution. Then a feature is randomly selected, and its label's position is randomly changed to another of the candidate positions. Then the change in the evaluation function is calculated. This change, written as  $\Delta E$ , is determined by how the changed position of the label affects the evaluation function, i.e. the general change in label placement quality of the map. If the new position lowers the total cost or if it does not change it, then the new label position is kept. But if this labeling position is worse than the previous, i.e. if the cost has increased, then there is a probability  $P$  that the new position will not be kept. The  $P$ -value is computed with *equation 2*.

$$P = 1.0 - e^{(-\Delta E/T)}, \quad \text{Equation 2}$$

where  $\Delta E$  is the change in the evaluation function (objective function) and  $T$  is the current temperature value. This procedure is repeated: the temperature is reduced; another feature is selected randomly; the label position of the feature is moved randomly to another candidate position; the evaluation change is calculated; the new position is kept or not, etc. (Edmondson et al. 1997). The utilization of the temperature parameter enables the algorithm of escaping local cost minima and finding better solutions, ideally the global minimum. The probability of escaping local cost minima decreases as the temperature decreases, which means that the escaping probability decreases for each iteration.

Christensen et al. (1995) found that the simulated annealing method produced good labelling results compared to other methods, while at the same time being relatively simple to use. Zoraster (1997) developed a similar combinatorial optimization algorithm, utilizing the same probability equation and temperature decrease as Christensen et al. (1995) and Edmondson et al. (1997). He aimed at making the algorithm suitable for more specific applications, for maps with dense point feature labelling and clustering of features, which can lead to the problem of extensive overlapping. Zhang and Harrie (2006a) applied simulated annealing in their method for real-time map labelling. Their annealing schedule was largely the same as Christensen et al. (1995) applied. The optimization method searches iteratively for solutions where the

objective function value is lower than the current solution value, and eventually settles at a local or global minimum solution (Zhang and Harrie 2006a).

There are other heuristic algorithms that can be implemented for finding optimal label placement solutions. Wagner and Wolff (1998) used a combinatorial approach with another heuristic algorithm, which consisted of rules for generating labelling and candidate removal. To decrease the number of candidate positions, candidates were deleted iteratively by prioritizing the points with the highest number of candidates left, and the candidates that had the most overlaps. A labelling solution was found when no point feature had more than one candidate label position. This algorithm was compared with a greedy algorithm method and a simulated annealing method. Greedy algorithms find solutions by always choosing the best option for the local situation. This greedy algorithm made a label candidate selection simply by always choosing the candidate position from the set that was furthest to the left. They found that the labelling produced by their new algorithm was of approximately the same quality as that of the simulated annealing method, while the greedy algorithm obtained poorer results (Wagner and Wolff 1998).

van Dijk (2001) also took a combinatorial perspective to label placement but instead used genetic algorithms. Genetic algorithms come from the idea that solutions can be developed and evolved much like biological entities in nature evolve over time, to reach optimal results (Russell and Norvig 1995). A population of solutions is created, and from this population, parents are chosen based on an objective function. These are combined to produce children that inherit the parents' properties. The children are added to the population, and the parents are removed. This process is repeated. The algorithm ends when a certain criterion has been met (van Dijk 2001; van Dijk et al. 2002).

### 3.2 Sequential placement

Labelling can also be automated by using sequential placement, where labels are placed in a sequence. Jones (1989) created a method for placing labels for point features, specifically names for towns represented by points. The program developed emphasized that names should be easily linked to their features, i.e. to minimize the ambiguity, and that label-feature overlaps should be avoided. Labels could however overlap other map features if the importance or color of the feature allowed it. Candidate positions for features were specified, and a favorable label position was chosen. A characteristic property of this method is that an

already placed label can change its position in order to give room for another label. If placing a particular label is problematic, then the last placed label can be moved to another of its candidate positions, so that a tolerable position can be found for the current label. This backtracking can occur in several steps when needed.

A sequential placement method was also applied by Cook and Jones (1990). They labelled point, line, and area features. Point and line features had 20 possible candidate label positions, while area features had less than 20 candidate positions. Possible label overlapping was stored in a list. Lists that define position prioritization were also produced for all features to be labelled, so that preferred label positions were chosen over other. The sequential labelling of Cook and Jones (1990) occurs as follows.

The feature that is the hardest to label, i.e. the one with fewest candidate positions, is labelled first. A feature can have few candidate positions due to removal of candidates that overlay other features or labels. The placement of a label hence often results in removal of other features' candidate positions. After the first feature has been labelled, the feature with the second lowest number of candidate positions is labelled, etc. Features with the same number of candidates are ordered by first labelling the one with the highest number of possible label overlaps. When a feature to be labelled has no candidate label positions available (due to feature or label overlapping), backtracking is performed. One or more of the most recently labelled features are relabeled in order to find a possible labelling solution. Cook and Jones (1990) effectively illustrate their sequential method with figures, and three of those are displayed in *figures 3.3, 3.4 and 3.5* below. They show a set of eight point features that are sequentially labelled.

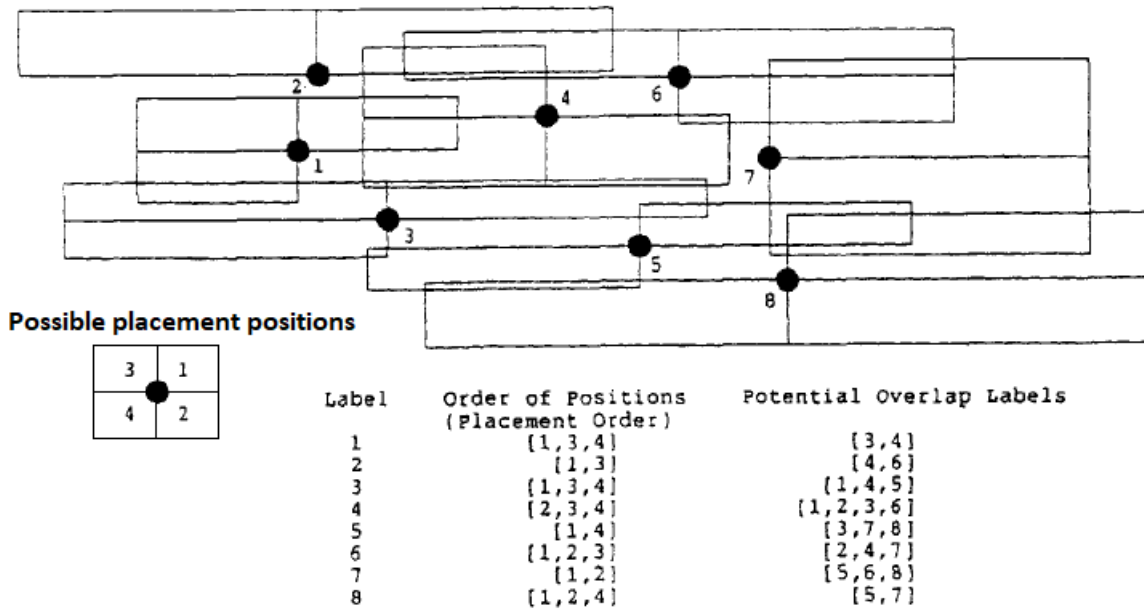


Figure 3.3: The initial state, where no labels have been set. The eight features' candidate positions are displayed as boxes. Four placement positions are used for simplification, where the number represents the placement priority. In the list, the available label positions for every label are ordered. The potential overlaps for every label are also listed. Modified after Cook and Jones (1990).

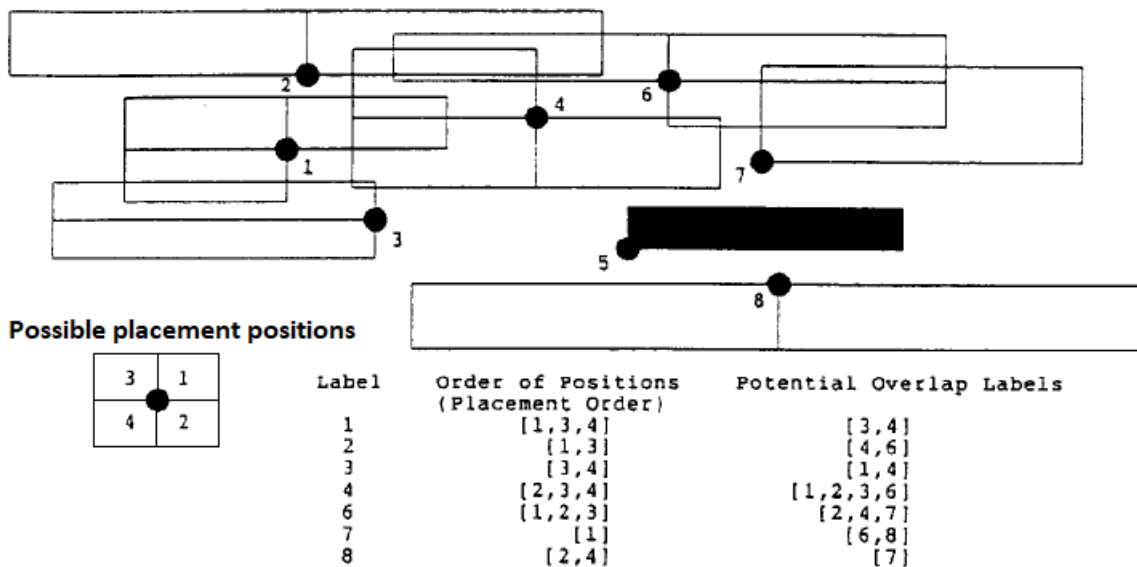


Figure 3.4: The first feature labelled is point number 5, since it has the fewest available positions (together with point 2 and 7), and the highest number of possible overlaps (together with point 7), as seen in the table of figure 3.2. The chosen label position is displayed as a black box. Modified after Cook and Jones (1990).



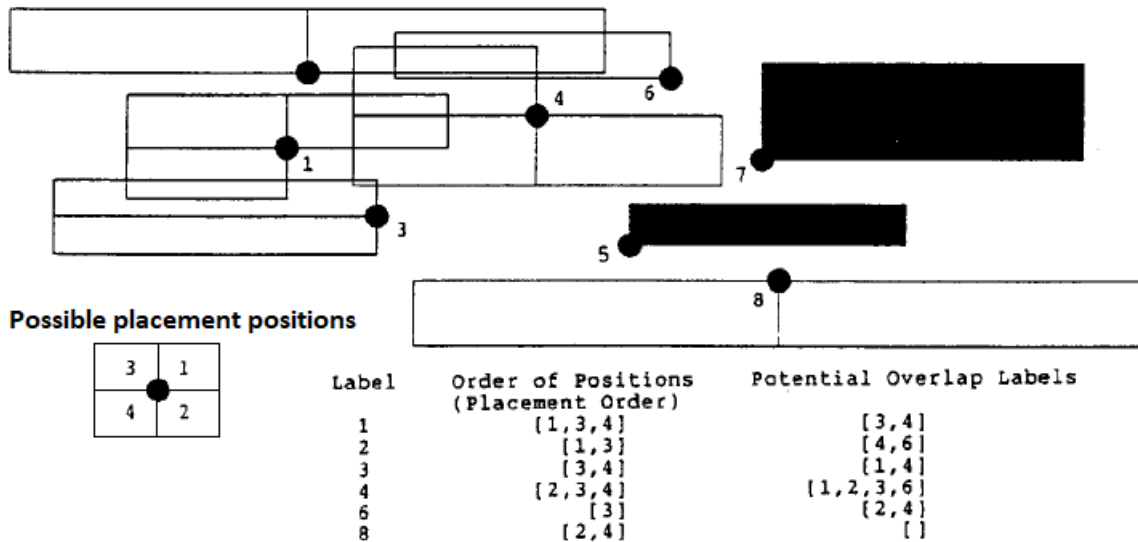


Figure 3.5: The next feature that is labelled is point 7 because it has the fewest available positions, as seen in the table of figure 3.3. Backtracking occurs if the next feature to be labelled has no available positions. The procedure continues until all features have been labelled. Modified after Cook and Jones (1990).

The sequential methods of these studies, where backtracking is applied in the label placement process, are also referred to as depth-first or exhaustive search algorithms in the literature, see e.g. Zoraster (1997) and Rylov and Reimer (2017).

### 3.3 Slider model

Another approach to automated map labelling is the sliding label method, which is based on the sequential placement method (van Kreveld et al. 1999). Here, the candidate label positions are not set at discrete positions like in the combinatorial optimization and the sequential placement methods. Instead, the labels can slide within defined boxes, which creates a higher number of possible positions for each label (van Kreveld et al. 1999). This method has mainly been investigated in point and line feature labelling (van Kreveld et al. 1999; Strijk and van Kreveld 2002; Zhang and Harrie 2006b). Sliding models from van Kreveld et al. (1999) are shown in figure 3.6. The boxes can be decreased in size if there are map features overlapping parts of them. Figure 3.7 illustrates an example of this.

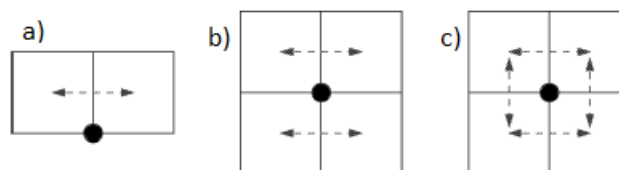


Figure 3.6: Three sliding models for point feature labelling: a) The label position can be adjusted horizontally within the box. b) Same as a) but with a box below the point feature. c) The label position can be adjusted both horizontally and vertically within the boxes. In all models, one of the boxes' sides must touch the point feature. Modified after van Kreveld et al. (1999).

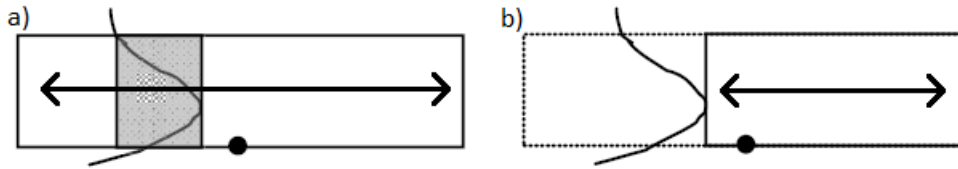


Figure 3.7: A box for point feature labelling. a) A line feature overlaps a part of the box (grey area). b) The box' size is decreased as the overlapped part of it is removed. Modified after Zhang and Harrie (2006b).

It has been suggested that slider models produce higher labelling quality than methods where labels are not able to slide (Strijk and van Kreveld 2002). van Kreveld et al. (1999) found that this method resulted in a fewer number of label suppressions. However, the slider method and the sequential placement method will not be dealt with in the practical part of this study, since PAL applies a combinatorial optimization method. The algorithms of PAL are described in section 4.2.1.

## 4. Label placement method and tools for practical study

### 4.1 Selection of method and tools

The selected GIS desktop application to use for the practical study of this thesis is QGIS version 3.12 (QGIS Development Team 2020b). QGIS is chosen because T-Kartor wish to investigate if they can utilize QGIS instead of ArcGIS for their map production. QGIS, previously known as Quantum GIS, is an open-source GIS software (Sherman et al. 2004). It is widely used in the GIS community. QGIS is written in the programming language C++ (Sherman et al. 2004). The programming language Python is also highly compatible with QGIS. Python is essentially object-oriented, meaning that a wide range of classes, methods, and functions can act as objects that interact in different ways with each other (Downey 2012). The labelling library PAL is integrated into QGIS, where it is used to label layers with point, line, and polygon features. Because of its accessibility in QGIS, and because it applies combinatorial optimization methods which parameters can be set, PAL is the chosen label placement algorithm library of this study. Moreover, Python is used for setting label placement parameters of PAL. This is because QGIS has an Application Programming Interface (API) for Python (Steiniger and Hunter 2013). This is known as PyQGIS. In the API, there are many classes for defining how labels are to be placed. Some of these are described in section 4.2.3. The coding of the Python program is performed in the *Code Editor* in QGIS, where one can open, edit, and run Python scripts directly in the QGIS application.

## 4.2 The labelling library PAL

PAL is a C++ programmed labelling library (Ertz et al. 2009). PAL is released under the GNU Lesser General Public License, which means that it can be used freely and together with other software. It consists of algorithms that creates labelling for point, line, and polygon features. Its method for label placement is a combinatorial optimization technique, which is described in greater detail below.

### 4.2.1 PAL algorithms

The PAL labelling library contains algorithms that use combinatorial optimization methods (Ertz et al. 2009). PAL utilizes an objective function that is defined by the sum of costs. The optimal solution has the objective function with the lowest summed cost. There are three costs types: placement, overlapping, and label suppression cost.

The two steps of PAL algorithms are candidate generation and optimization, as shown in *figure 4.1*. First, candidate label positions for a feature are produced. They are generated differently for point, line, and polygon features. A placement cost is computed for each candidate. The cost is determined by (1) label placement in relation to the feature; (2) eventual overlapping of another map feature, or the distance to another map feature. There is also a third cost for label suppression. The placement costs are computed differently depending on feature type. The creation and placement of label candidate positions also varies depending of the placement option selected. In QGIS version 3.12, there are three placements options for generating label candidates for point features: *Cartographic*, *Around point*, and *Offset from point*. The placement options for line feature candidates are *Parallel*, *Curved*, and *Horizontal*. For the *Parallel* and *Curved* options, one can select whether the candidates should be placed *Above*, *On*, or *Below line*, or have a *Line orientation dependent position*. There are six placement options for polygon feature labels. Two of those are centroid based: *Around centroid* and *Offset from centroid*. With the options *Using perimeter* and *Using perimeter (curved)*, the perimeter of polygons are treated as lines which labels follow. Candidates can also be placed inside polygons, either tilted with the *Free* option, or horizontal by using the *Horizontal* option (QGIS Development Team 2020b). The label candidate placement cost for polygons increases with decreasing distance to the boundary of the polygon or to other features. Of the many candidates produced for a feature, a portion is removed due to bad placements. These can be label candidate positions causing overlapping, and other positions with high costs (Ertz et al. 2009).

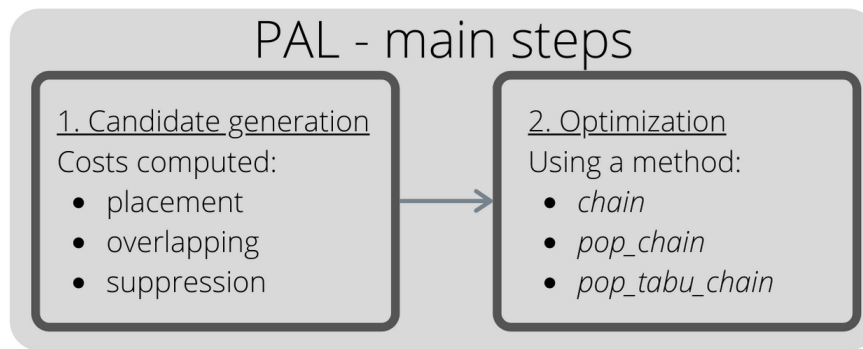


Figure 4.1: In PAL there are two main steps of the labelling: candidate generation and optimization.

In the next step, the optimization, it is decided where the labels should be placed, choosing from the candidates associated to each feature. Every candidate position has information stored on which other candidates that it creates overlapping with, i.e. the candidate overlaps are stored. This is because such overlapping is prohibited in the labelling. If a feature cannot be labelled due to overlapping, a cost is applied to it for penalizing its inability to be labelled. This is the label suppression cost. Cost values range from 1 to 10. The cost depends on the priority of the layer that the feature belongs to. The cost increases with increasing layer importance. Features thus have costs for placement, overlapping, and suppression. The algorithms output a labelling solution with a low total cost by maximizing the number of labels present and by selecting good label placements for the features (Ertz et al. 2009).

An initial solution is modified to optimize the label placement, by applying one of the optimization methods described by Ertz et al. (2009). The *Chained neighbourhood (chain)* method is the simplest and fastest of the optimization methods. It runs a chain of label placement modifications. It allows no label overlapping. A feature is relabeled to the next optimal candidate position. If this increases the quality of the overall labelling, then the position is kept, and that labelling version is chosen to be the optimal up to that point. Then another feature is relabeled, etc. However, if the relabeling results in one label-label overlap, the other label is repositioned, etc. But, as a worst-case scenario, if the relabeling results in a higher number of overlaps, or if there is only one overlap but the other label cannot be repositioned, then those labels are suppressed. Here the improvement iterations end, and the most optimal labelling found in the chain is chosen to be the final labelling solution. A feature can only be relabeled once in the chain (Ertz et al. 2009).

The second optimization method is called *POPMUSIC with chain (pop\_chain)*. POPMUSIC is a metaheuristic technique for finding optimal solutions on a local scale by improving smaller, separate parts of a solution problem (Taillard and Voss 2002). In *pop\_chain*, this

method is combined with the *chain* method. A chain is run, and the last solution from that chain is kept if the chain contains a solution that is better than the current solution. Then a new chain is run, starting with a different feature, etc. The iteration ends when chains have started on every feature. The third method option, *POPMUSIC with tabu search and chained neighborhood* (*pop\_tabu\_chain*), is a variation of the POPMUSIC method, where a tabu search frame is applied in the chain. It is called tabu because reversed label placement changes are not allowed in the chain. This is to avoid being iteratively trapped in the same solutions. Placement changes are made locally, and solutions which decrease the labelling quality are also allowed. But it is the most optimal, valid change that is chosen at each step in the chain (Ertz et al. 2009).

#### 4.2.2 PAL implementation in the QGIS user interface

In the QGIS graphical user interface (GUI), labelling parameters of PAL can be used in an easily accessible manner. There are many different tools for creating and editing labels. For each layer, one can set up customized label settings. There are submenus concerning label text, formatting, placement, shadow, etc. In this section the labelling functions and options of PAL in QGIS are reviewed.

In QGIS, each layer has a *Labels* tab in the *Layer Properties* window. In the *Labels* tab, one chooses between *Single Labels*, *Rule-based Labeling*, *Blocking*, and *No Labels*. Under *Single Labels*, the attribute field to produce labels from is chosen. In the *Single Labels* window there are several submenus. In the submenu *Text*, the font, style, size, and color of the label text can be specified, among other text properties. Below in *figure 4.2*, the submenus are visible to the left. Under *Placement*, one can customize the label placement of the selected layer by setting parameters. The *Placement* parameters are different for point, line, and polygon layers. In *figure 4.2*, some of the labelling parameters for a line feature layer are shown, e.g. the *Parallel*, *Curved*, and *Horizontal* label position options for a line feature.

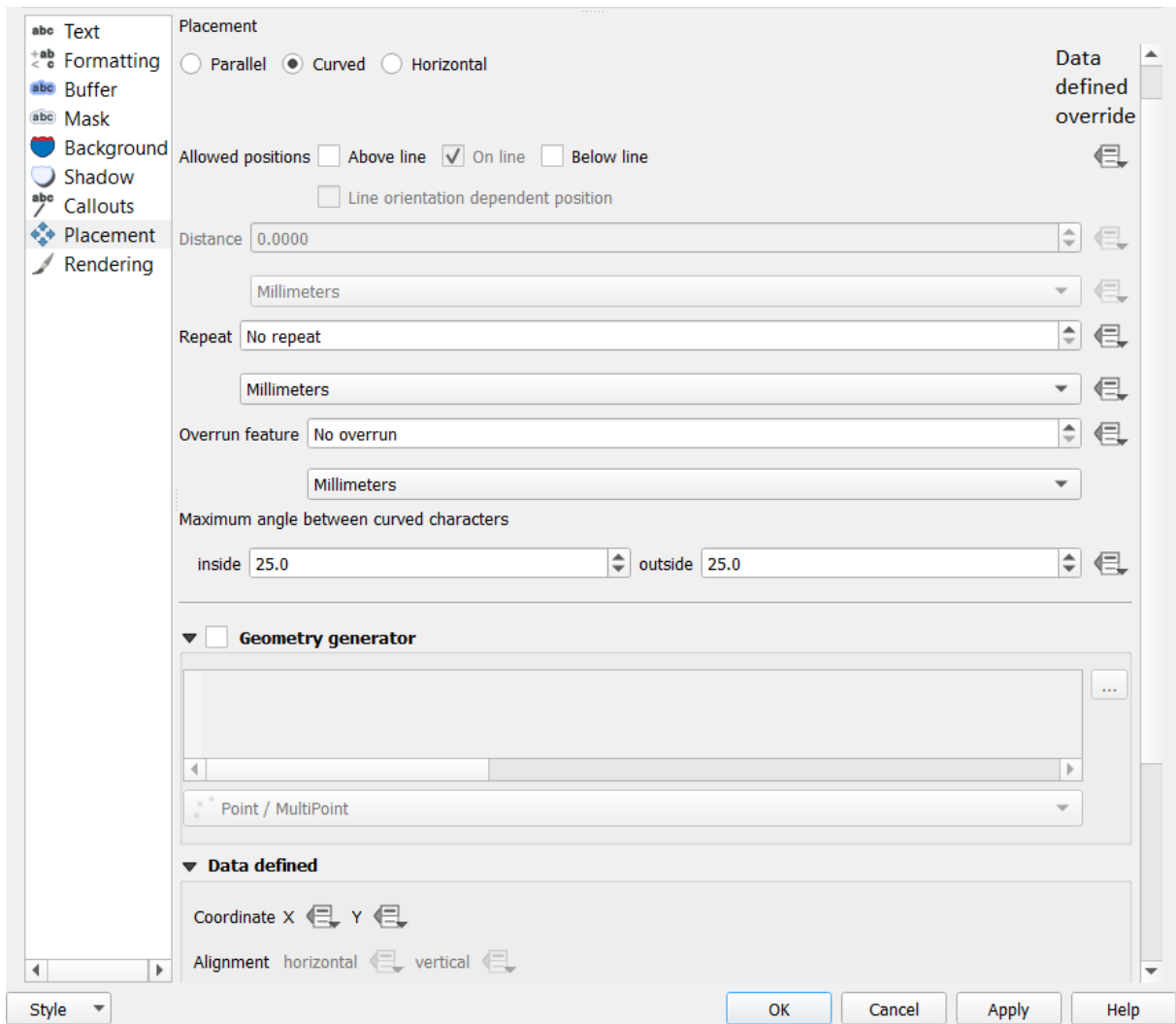


Figure 4.2: A part of the label placement menu for line features in QGIS. One can e.g. choose if labels should be placed above, on, or below their line features. Modified after QGIS Development Team (2020b).

For most parameters, one can edit the setting using the *Data defined override* button found on the right-hand side of the parameters in the label menu. The buttons are displayed in figure 4.2, e.g. beside the *Allowed positions* and *Repeat* parameters. *Data defined override* can be activated, and editing is made in the *Expression String Builder*. Here, customized expressions can be created for setting parameters such as *Size*, *Repeat*, and many more. Attribute fields and values in the layer, operators, math functions, and more can be accessed to build expressions.

QGIS has an *Automated Placement Engine* where one can specify automated label placement settings for all layers. It is found in the *Labels* tab, via a button in the upper-right corner. Clicking the button opens the *Automated Placement Engine* window, shown in figure 4.3. In the window a few parameters can be set, e.g. the number of candidate label positions for line and polygon features, per cm and cm<sup>2</sup>, respectively. In previous QGIS versions, e.g. in

version 2.4, the *Search method* parameter was included in the *Automated Placement Engine* window interface. *Search method*, also called optimization method, is the solution improvement technique used by the algorithm after an initial label placement solution has been computed. Five different search methods have been available for selection in previous QGIS versions. Apart from the three methods described in section 4.2.1 (*chain*, *pop\_chain*, and *pop\_tabu\_chain*), *pop\_tabu* and *falp* were available. However, their functionality is not well documented. The default search method used by the *Automated Placement Engine* is the *chain* method, and the default number of line candidate labels are 5 per cm and polygon candidate labels are 2.5 per cm<sup>2</sup> (QGIS Development Team 2020b). However, the *Search method* parameter is not available in QGIS 3.12, which is used in this study. The *Automated Placement Engine* can thusly be used to set algorithm parameters described in section 4.2.1 to generate label placement for all layers with activated labels, though it is important to note that not all algorithm parameters are available with *Automated Placement Engine*. An example of label placement utilizing the *Automated Placement Engine* is found in *figure 4.4*, where the settings displayed in *figure 4.3* have been applied. The boxes are the candidate label positions of the features: The empty boxes are candidate positions that were not selected. For example, the polygon label *Rosewood London* has 4 candidate positions.

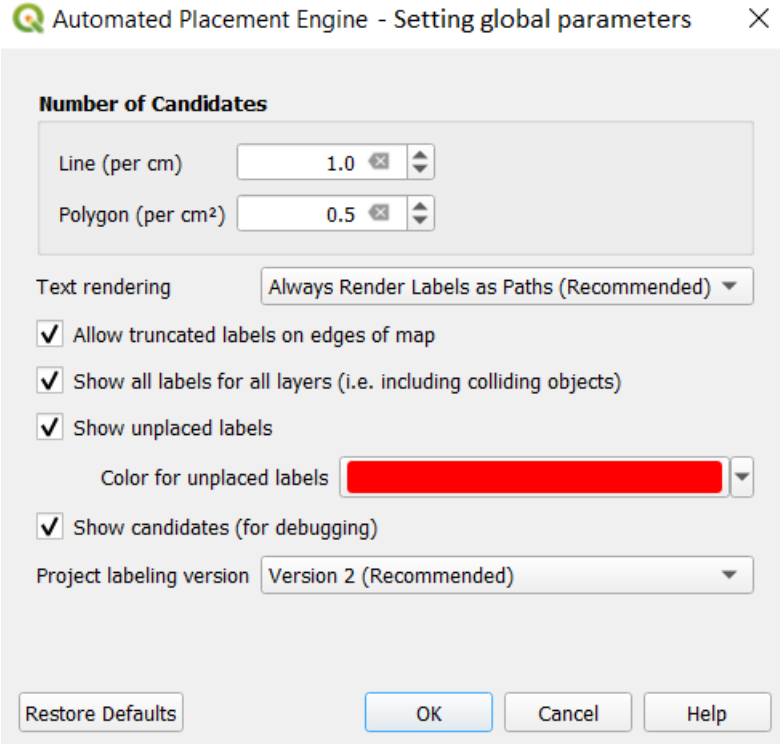


Figure 4.3: The Automated Placement Engine window. The program executes label placement for all vector layers with labels activated. Modified after QGIS Development Team (2020b).



Figure 4.4: Example of polygon and line feature label placement using the Automated Placement Engine. Unplaced labels are in red. The candidate label positions are shown as boxes. Note that the road labels have a box for every character. This is because they are set to be curved. Straight (parallel) labels have one box per label. Based on data from © T-Kartor Group AB.

The *Rule-based Labeling* selection has the same labelling options as *Single Labels*, but rules can be set for how the labels should be rendered. For example, one can set scale-dependent labelling. The *Single Labels* and *Rule-based Labeling* is designed for text labels: They have quite limited options for symbol or icon labelling. The *Blocking* option inactivates labels for that layer, and labels of other features are discouraged, or blocked, to overlap the features of that layer. Lastly, the selection *No Labels* simply inactivates the layer's labels.

Scalable Vector Graphics<sup>1</sup> (SVG) and other types of symbol and icon labels are more easily rendered using the *Symbology* tab, where the symbology of a feature is defined. For example, an *SVG marker* symbol can be selected as *Symbol label type* for point feature labels. Technically, this means that the symbol label is rendered as a feature. Another tool that can be used for label placement in QGIS is the *Geometry Generator*. With this tool one can build unique expressions to create or alter geometries. Hence the position of labels could be defined using the tool. *Geometry Generator* is available for both labels and their features, i.e. in the *Labels* and *Symbology* tabs under *Layer Properties*.

<sup>1</sup> <https://www.w3.org/TR/SVG2/>



### 4.2.3 QGIS Python API for labelling

The object-oriented programming language Python is largely based on modules, classes, methods, and functions that can be used to define and change objects and making them interact in a computer program (Oliphant 2007; Downey 2012). Modules store collections of classes and functions (Oliphant 2007). An example on a module is the *qgis.core* module, which is used to access PyQGIS classes needed to process data in QGIS. Python objects are defined by utilizing classes. An object can be created by defining an instance of a certain class, in the form of a variable. Classes can be used by invoking one of their methods. A method is a type of function that belongs to a certain class (Downey 2012). By using methods, different kinds of information can be defined, returned, and computed.

There are many PyQGIS classes and methods that relates to labelling layers in QGIS. All of these are found in the complete API documentation (QGIS Development Team 2020a), of which some are reviewed here. There are classes for setting parameters of the labelling engine in QGIS, e.g. the *QgsLabelingEngineSettings* class. By using it, global PAL algorithm parameters such as search method, number of candidates, and drawing settings can be set by invoking the corresponding methods. For example, *setMaximumLineCandidatesPerCm* is a method used to set the number of label candidate positions per cm for line features. This class thus relates to the functionality of the PAL algorithms described in section 4.2.1. It is the API corresponding to QGIS' *Automated Placement Engine* described in section 4.2.2, but by utilizing the class, a few more labelling parameters are available, e.g. search method, the color of unplaced labels, and reading settings from a project (QGIS Development Team 2020a).

There are also classes for setting the label placement options found in the QGIS interface, under the *Labels* tab. By using *QgsPalLayerSettings*, a large number of labelling properties for a specific layer can be set. With the method *placement*, the general label placement of a layer is set. For example, if it is set to 0, polygon labels are placed *Around centroid*. The class *QgsPalLayerSettings* can be used in conjunction with the *QgsProperty* class to set label properties as data defined properties for a certain layer. This is the API equivalent of utilizing the *Data defined override* functionality explained in section 4.2.2. Label size, rendering options, alignment of multi-line text labels, and much more can be set by applying those classes. The class *QgsTextFormat* is used to set text formatting options, e.g. the color and font style of text labels. To set symbology options for SVG, the *QgsSvgMarkerSymbolLayer* class can be applied (QGIS Development Team 2020a).

#### 4.2.4 PAL applied in research

There has been research conducted where PAL labelling library has been used to study label placement quality. Ertz et al. (2009) compared the PAL optimization methods described with respect to the proportion of labels displayed and the computational time. They tested the methods using *RandomRect* and *HardGrid*, which are two types of problem instances, described by Wolff (2002). *pop\_chain* and *pop\_tabu\_chain* were able to display more labels than the other methods available in PAL, but they were also the methods that needed the most processing time. A simulated annealing method was also tested, and it outperformed the other methods in the *HardGrid* problem, while not achieving as good results in the *RandomRect* problem. The tested methods that are not from PAL are described by Wolff (2002). The results are displayed in *figure 4.5*.

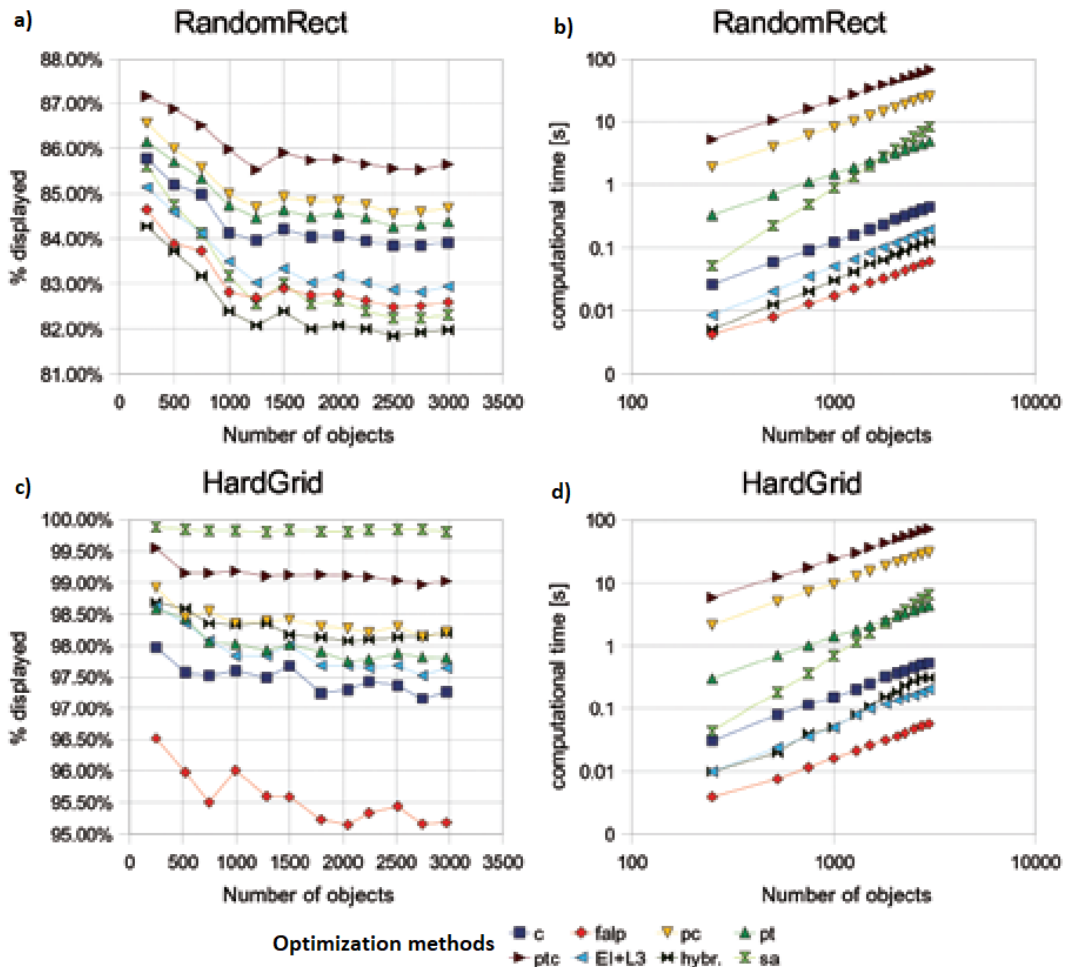


Figure 4.5: Eight optimization methods compared by applying two different label placement problems. a) Percent of labels displayed in the *RandomRect* problem. b) Computational time of the labelling in the *RandomRect* problem. c) Percent of labels displayed in the *HardGrid* problem. d) Computational time of the labelling in the *HardGrid* problem. The optimization methods are chain (c), FALP (falp), *pop\_chain* (pc), *pop\_tabu* (pt), *pop\_tabu\_chain* (ptc), edge-irreducibility algorithm (EI+L3), hybrid algorithm (hybr.), and simulated annealing (sa). Modified after Ertz et al. (2009).

Laurent et al. (2009) applied PAL algorithms, optimization techniques such as POPMUSIC to produce point, line, and polygon labelling. They found that three different POPMUSIC optimization methods produced relatively good results, compared to other optimization techniques. In a more recent study, Klute et al. (2019) tested automated labelling in QGIS using PAL. Afterwards, they made manual adjustments. They call this approach human-in-the-loop labelling. After automated labelling had been performed, using one of four algorithms, manual changes were done. This included label position changes, removal of point features and label candidates, or changing the size of text, among other things. Then, an algorithm was selected to continue finding a new labelling solution, etc.

## 5. Practical study of label placement

In this section the practical part of this study is described. The labelling rules followed are reviewed. The workflow of the practical study and the data used are stated. Then the design of the Python program is described regarding the order of the label placement and the overall implementation.

### 5.1 Labelling rules currently applied by T-Kartor

Åsa Nilsson, T-Kartor's administrator for this project, was interviewed on how the company creates good label placement for their maps: what techniques, methods, and rules they apply. The interview took place at T-Kartor's head office in Kristianstad, Sweden, on the 25<sup>th</sup> of February 2020. It was conducted in Swedish. The whole interview was recorded with a mobile device. Notes on the answers were also taken during the interview. After the interview, the recording was transcribed, and additional information found from the transcription was added to the interview answers. When the interview answers had been compiled, they were sent to the interviewee, who gave comments on the answers. The interview answers were then revised based on the comments. Then, a summary of the information obtained from the interview was created. The summary was then translated to English. A comprehensive description of the interview in Swedish is given in *Appendix B*.

This section contains the summary of the information received from the interview, but also information from meetings held at T-Kartor, and from additional data provided by the company. This information concerns the labelling rules and principles that the company applies to produce maps with good cartographic quality. More specific descriptions on e.g.

label sizes and colors are described in design standards produced by Transport for London (2009; 2011), which is a transport authority serving Greater London. This information is not presented here, but their standards are used as a reference for this project, and for T-Kartor's map production in general. Furthermore, T-Kartor's labelling rules are compared with the literature on labelling, and the labelling rules applied in this study are discussed.

Below the rules for point, line, and polygon labelling are described. It should be noted that there are also rules that are applicable for all label types. First and foremost, legibility is the most important quality of good labelling. Ensuring that one can easily tell what map features the labels are referring to, so that there is no ambiguity. This is to make it possible for the reader to find places via the map.

#### 5.1.1 Point feature labelling

Point feature texts should always be placed horizontally. For point feature text placement, the order of prioritization is as follows: (1) To the upper left of the point; (2) To the upper right of the point; (3) To the lower left of the point; (4) To the lower right of the point. However, this rule is not particularly relevant for text labels, since most texts belong to roads and landmark buildings, none of which are point features. Points are usually represented by icons and symbols, rather than by texts. Virtually no point features are represented by an actual point on the maps. Instead, the point feature's symbol or icon is placed on the relevant spot; alternatively, in exceptional cases, the symbol or icon is placed at a smaller distance from the spot, with a leader line showing the location of the place. This labelling approach is demonstrated in *figure 5.1* below.

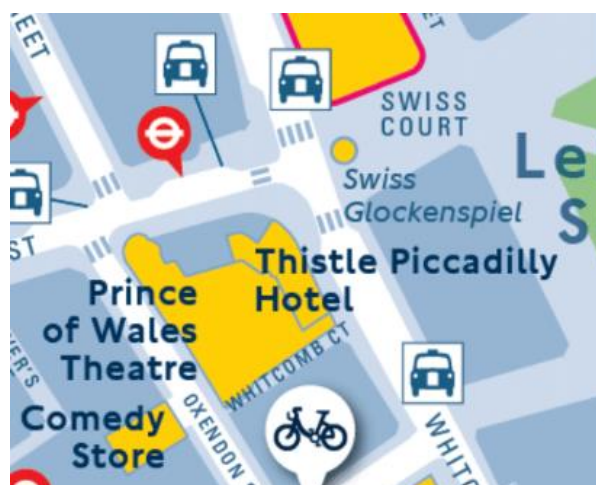


Figure 5.1: The taxi symbols are an example of point feature label placement, where the point is not visible, and the symbol is placed on the point location, alternatively a leader line is used to point on the point location. From © T-Kartor Group AB.

### 5.1.2 Line feature labelling

Road name labels are always placed within the borders of their corresponding road, which means that they must adhere to the shape of the road. Texts are clearer when they are straight. Therefore, straight line feature labels are preferred. But if the road has a significant curve, the label cannot be entirely straight, since it needs to adhere to the shape of the road. However, roads with only a slight curve can have straight labels. If a text label does not fit within its road, it should be placed across two lines within the road, if possible; if not, the label should be shortened. A label is repeated on the line if the road is quite long. Road labels should be placed at the center of the road, and not at the end of the road. Line feature texts should never end at an intersection, since one cannot tell whether the road continues after the intersection, or if another road takes start. For this reason, the label should be placed across the intersection. It is good if labels of parallel roads are aligned in a row, as the road labels in *figure 5.2*.



Figure 5.2: An example of how road labels can be placed. From © T-Kartor Group AB.

The line features that should be labelled with text are roads and creeks. They all have similar rules for placement, as described above. Squares can also be considered line features, and their labels are placed horizontally across the square. The Thames is technically considered to be a polygon feature, rather than a line feature. The text label for the Thames is usually straight. It may however be curved in certain situations. The text follows the direction of the river. It is always in capital letters. Zebra crossings should end with gaps on both sides of the road. Road labels should avoid overlapping zebra crossings.

### 5.1.3 Polygon feature labelling

Preferably, polygon labels should stay within the boundaries of their polygon features, but they may cross them if necessary. However, this should preferably be avoided by stacking the

text. Polygon labels should be horizontally placed and aligned depending on their placement in relation to the polygon feature they belong to. Generally, if a label needs to be placed more to the right of its polygon, then left alignment should be used, and vice versa. The aligned part of the text, or at least a fraction of it, should always be inside the polygon. If a landmark building label overlaps a road, its color should be changed from dark blue to white. All these principles are applied in the polygon labelling illustrated in *figure 5.3* below. The order of prioritization for point labels described in section 5.1.1 can also be applied to polygons and their centroids, to label landmark buildings with texts and symbols.



*Figure 5.3: A good example of polygon feature label placements. The yellow landmark buildings are labelled with dark blue or white texts. From © T-Kartor Group AB.*

#### 5.1.4 Symbols and icons

Cycle hire symbols should preferably be placed in a 90-degree angle to its corresponding road. Other angles are permitted if necessary, but the cycle hire symbols should not be placed over roads. However, they can be placed over roads if it is needed in certain situations. Symbols are also permitted to overlap buildings when necessary. The bike and traffic roundel symbols, e.g. the roundel of the bus stop symbol, must always be horizontally placed, but the circle that encloses them may be rotated as needed. The tip of the symbol arrow, i.e. the built-in leader line of those symbols, must point directly at the location. Thus, the symbol may only be rotated around the point. Bus stop and cycle hire labelling is shown in *figure 5.4*. Symbols such as bus and taxi symbols must be placed in their determined position. Bus station symbols may not be placed on roads.



Figure 5.4: Bus stops and cycle hire stations labels with symbols. Note that some of the labels overlap roads, which is permitted if no better solution is found. From © T-Kartor Group AB.

Retail unit icons, toilet icons and station symbols are initially represented by a number attached to the rest of the text, as a single string of text. The numbers are switched to their corresponding icons in the post-processing. Station symbols should always be placed on their point or polygon, while the station text may be placed somewhere around its symbol, to ensure an optimal placement solution. However, the station symbol must always be placed on the feature it is representing. The station text should be aligned with the symbols, both in the x- and y-direction. This problem must be fixed in the post-processing.

Texts, symbols, and icons have different alignment depending on the situation: they could be centered, or they could be left or right aligned. Retail unit icons are always placed in their own row, under the text. They are usually centered, but not always. Landmark buildings may have retail unit icons, those are left aligned. Neighborhoods, however, have their retail unit icons centered. There is no upper limit for the amount of icons attached to a single point, since the number has been predefined for each location. However, no more than four icons are displayed for each location. Unlike retail area icons, toilet icons can be placed either on one or two lines. *Figure 5.5* exemplifies how retail area icons can be placed.



Figure 5.5: Placement of retail area icons. From © T-Kartor Group AB.

#### 5.1.5 Label overlapping and removal

No texts, symbols or icons may be removed or suppressed. Labels are edited and repositioned in the post processing in order to produce satisfactory labelling. When necessary, a leader line can be made, pointing to the symbol or icon's correct location. It is mainly only the large, transparent neighbourhood, borough, and village labels that may overlap other information on the map. An example of this can be seen in the upper right parts of *figure 5.3*. It is allowed for an unrelated text to partially cover a building, if it is clear which building each text corresponds to, and that the building's own label is clearly visible. This is the case in *figure 5.1*, where the building label *Thistle Piccadilly Hotel* partly covers the neighboring building *Prince of Wales Theatre*. Cycle hire symbols and similar symbols and icons are also allowed to overlap buildings if necessary. Since texts may not be removed, icons and symbols may sometimes be allowed to cover texts. Normally, a label should not cover an icon, but this is allowed if it is the best solution. It is preferred that roads have no labelling other than their road labels.

#### 5.1.6 Additional rules and considerations

The vector layers for the base map always have a set drawing order, but it is not necessarily the same for texts, symbols and icons. They may be rendered in a different order. Texts should be placed after symbols and icons have been placed, because the position of the texts should be adjusted in relation to the symbols and icons. There are rules to how labels are shortened. For instance, the definite article *the* may not appear alone on a line. Proper nouns, such as *Hotel*, may stand by themselves on a line, as in *figures 5.1* and *5.3*. The preposition *of* needs to be preceded by a new line. If a label does not fit, it should be shortened, or stacked in several lines. The scale of the maps is 1:2,250.



The You Are Here (YAH) point is always placed one third from the bottom and two thirds from the top of the map. Buildings and texts may sometimes be removed from the map if they are far from the YAH point or if they are at the edges of the map, as they are less important than other information on the map. The information in front of the map reader, i.e. in front of the YAH point, is the most important. No texts, symbols or icons may overlap the YAH circle, which encloses places that can be reached with a 5-minute walk. The placement of the YAH circle necessitates repositioning of many labels in the post processing, as they overlap the circle.

#### 5.1.7 Comparison with the labelling rules from the literature

The labelling rules currently applied by T-Kartor are quite different than labelling rules from the literature, which are reviewed in section 2. This is partly due to the difference in scale. Many labelling rules reviewed in section 2 concern small scale maps, while T-Kartor's Wayfinding Maps are in larger scale. However, some fundamental principles are the same. For example, the legibility of labels should be as high as possible, and the ambiguity of the feature-label relationships should be minimized. Furthermore, horizontal and straight labels are preferred over tilted and curved labels for describing point features. Another common rule is that labels describing lines should be placed along the lines, so that they follow the shapes of the line features. Overlaps of labels and features should be avoided.

Despite the similarities described above, there are some major differences between the rules that T-Kartor applies and general labelling rules. In contrast to the literature, which says that point feature labels should be placed around the feature, most point feature labels on the Wayfinding Maps are placed on the point location. This is especially true for many of the symbol labels, e.g. taxi ranks, ticket stops, and post offices. However, the cycle hire and bus stop symbols have leader lines that point to the point feature location, which means that they technically are placed around the feature. Concerning road labelling, T-Kartor places road labels on the roads. The literature says that line feature labels most often are placed above or below the line (Robinson et al. 1984; Slocum et al. 2005). However, the roads on the large-scale Wayfinding maps are polygons, rather than line features, although the road labels belong to line features that are not rendered on the maps. Therefore, polygon labelling rules can be applied on road labelling.

Slocum et al. (2005) state that polygon features that are not horizontal in their shape should have their labels tilted to follow the shape of the polygon. T-Kartor however have different rules for different polygon types. Road labels should follow their road features. All landmark building labels, no matter the shape of the building, should be horizontal. The River Thames-polygon mostly has straight, tilted labels, but they can also be curved or horizontally placed if appropriate, e.g. due to the shape of the river. These are just a few examples of how T-Kartor's labelling principles differ from other sources. There are other examples that are not brought up here. Further labelling information concerning the City Wayfinding maps can be found at Transport for London (2009; 2011).

#### 5.1.8 Labelling rules applied in this study

Some of T-Kartor's label placement rules are the same or similar to labelling rules found in the literature. However, as stated in section 5.1.7, there are many cases where rules differ considerably. In this study, the labelling rules currently applied by T-Kartor are exclusively followed. This is motivated by the fact that the labelling of the city maps is ultimately meant to satisfy the company's standards concerning the placement of labels. The analysis and evaluation of the labelling is also based on the company's rules on label placement.

## 5.2 Workflow

T-Kartor's labelling rules that are summarized in section 5.1 is used as a reference for developing a Python program for label placement. In the program, PAL-parameters are set. T-Kartor provides test data, on which the label placement program is run. Maps with different labelling parameter settings are produced, which have different label placement. The maps are in PDF-format. In the post-processing, misplaced labels are repositioned and edited manually in Adobe Illustrator. However, the post-processing of the maps is not within the scope of this thesis.

Parallel to this study, Wei (2020) evaluates the labelling quality of a map produced by the label placement program and PAL. He does this by adding code that simplifies text labels into bounding boxes. However, the maps produced for this study does not have simplified labelling (see *figure 5.9*). In the simplification done by Wei (2020), road labels are represented by red bounding boxes and landmark building labels are represented by black boxes. The symbol labels remain unchanged. The simplified output map is split into 35

smaller simplification images in raster format. One of the simplification images is seen in *figure 5.6*.



*Figure 5.6: One of the simplification images that is used as input in the quality level schema applied by Wei (2020). The road labels are represented by red bounding boxes and the landmark building labels are represented by black bounding boxes. The symbol labels are not changed. From Wei (2020), based on data from © T-Kartor Group AB.*

The label placement evaluation performed by Wei (2020) consists of a quality level schema used to grade the quality of the label placement in a quantitative manner. His quality function is based on three criteria: legibility, disturbance, and association. The input to this evaluation is a simplification image. The quality function grading is done for each individual label in the image. Each label's placement is graded as (1) bad, (2) moderate, or (3) good, with respect to each of the three criteria. A label is given its lowest grading of the three criteria, and an image is assigned the same grading as the most poorly graded label in that image. Icon and symbol labels are not analyzed in the evaluation other than as obstacles. The output of the quality function is a grading of the whole image. Wei (2020) applies artificial intelligence (AI) for the quality evaluation of images. He uses the 35 simplification images for testing the AI program, and 2400 images for training the program. The label placement evaluation using the quality level schema and AI is solely done by Wei (2020).

The grading of images is meant to provide feedback to this study. However, no such feedback is used to improve the label placement program. This is because the AI is not performing in a desirable fashion (Wei 2020). Still, the combined workflow of this study and Wei (2020) is visualized in *figure 5.7*. This is the desired workflow for evaluating labelling and improving PAL parameter settings. The parts belonging to this study are on the left side of the dashed, orange line. Wei's (2020) parts are on the right side of the line. However, the actual workflow that is applied in this study is different. Instead, maps are visually examined. This provides feedback to the label placement program, where parameter settings are improved. Then new maps are produced, etc. In *figure 5.8* this smaller workflow is presented.

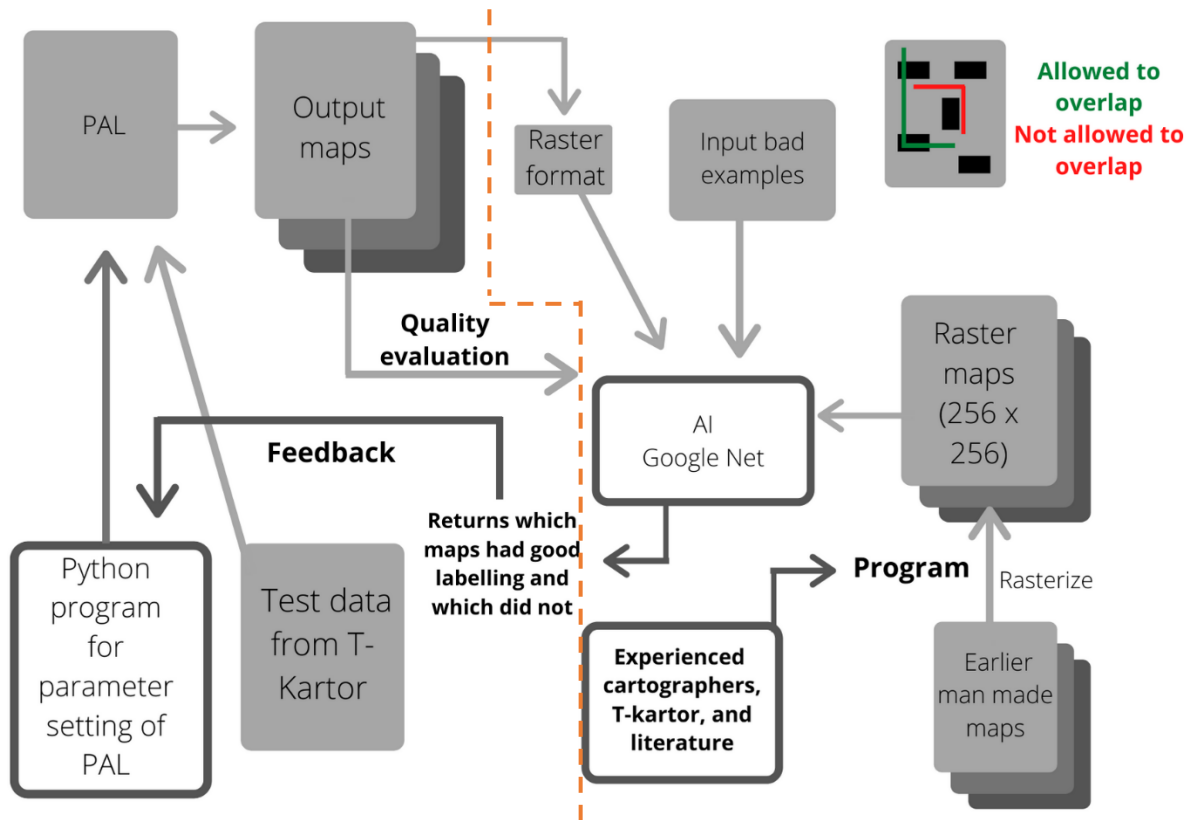


Figure 5.7: The desired workflow of this study, visualized in the left part of the figure, and of Wei (2020) in the right part.

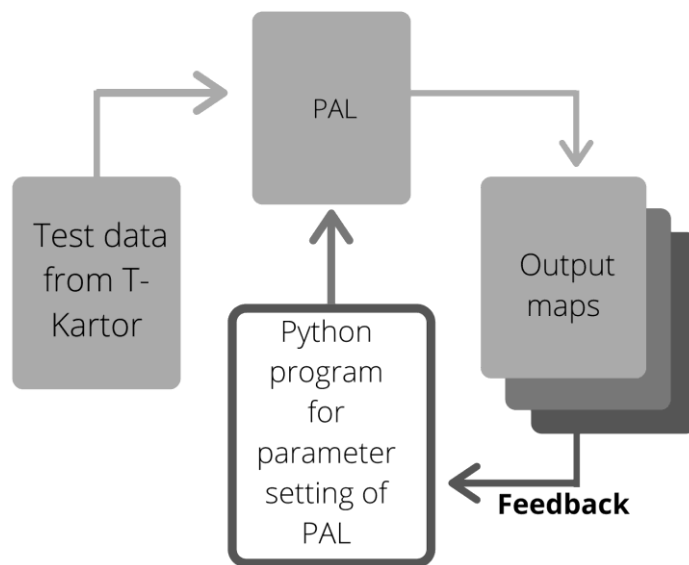


Figure 5.8: The applied workflow of this study.

### 5.3 Data

The data provided by T-Kartor are a QGIS Project file (*London.qgz*), a QGIS Layer Settings file (*LMF.qml*), and a GeoPackage (GPKG) file (*LMF.gpkg*). The QGIS Project file contains

vector layers with features: points, lines, and polygons in GPKG-format, illustrating an area of central London. The vector layers have various attribute data associated with them, e.g. the names of roads, stations, neighborhoods connected to features, and other information. The text label data are stored as attributes for the features. Furthermore, SVG and TrueType (a digital font format) files are provided for usage for symbol and icon labelling. The SVG files are saved as *Optimized SVG* file format for good graphical presentation. OpenType fonts, which are extensions of the TrueType format, were available for the text labels: Univers 57 Condensed, Univers 67 Bold Condensed, and New Johnston (Bold, Medium, and Light). The path to the SVG files is defined in the QGIS application, and the fonts are installed on the computer. Transport for London (2009; 2011) has specifications for the different map elements, for properties such as colors, fonts, sizes, symbols to be included, etc. Thereby the maps produced should follow a certain design standard.

#### 5.4 Label placement program

Here follows the methodology of the label placement program development. The Python label placement program is written in the *Code Editor* in QGIS 3.12. It sets PAL label placement parameters, so that the label placement produced should meet the rules described in section 5.1. The main parameter settings of the script are described below. The whole program code can be found in *Appendix A*.

The API for layer specific labelling parameters are applied in the label placement program developed. The layer specific parameters are set so that the labelling follows the rules to the largest possible degree. Many parameter values are set empirically to find suitable values, e.g. for label repetition, label blocking, and label alignment. The global *Automated Placement Engine* parameters are set manually. Since the symbol labels are not rendered as labels but as features, the global settings do not affect them; it only controls the text labels. The different types of map labels are added, layer by layer. *Figure 5.9* shows a flowchart of the label placement program.

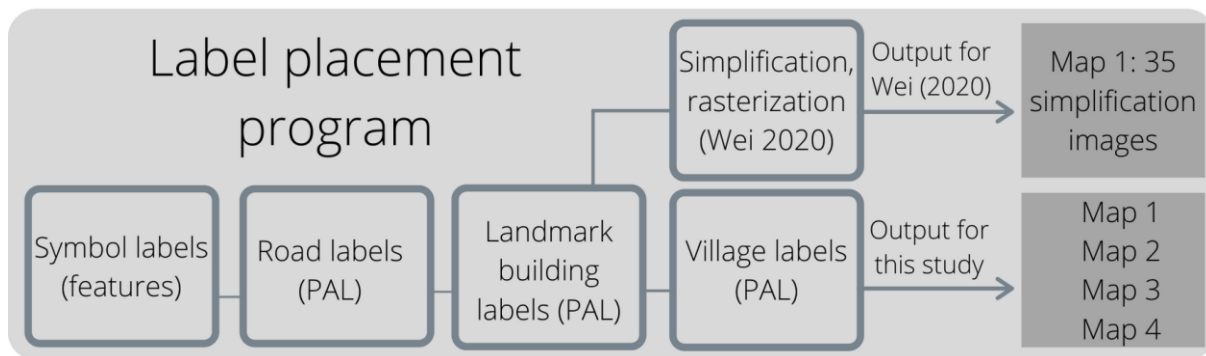


Figure 5.9: A flowchart of the label placement program. It shows the structure of the program, and how the output for this study and for Wei (2020) was produced.

The symbol and icon labels are rendered first for presentation purposes: They are point feature labels that are to be placed at set positions, i.e. directly on their point location. The symbols that fall into this category are car parks, building entrances, ticket agents, taxi ranks, station entrances, post offices, police stations, and info centers. All symbols, except for the info center symbol, are rendered as features with *SVG marker* symbol layer type. The info center symbol is also a feature, but it is rendered using *Font marker* instead. To reduce overlapping, label blocking is set for all symbols. This means that other labels are not as likely to overlap the symbols. The obstacle weight is set to 9.5 on a scale of 0-10, where higher values decreases the chance of overlapping.

The first text labels that are rendered are the roads. The road labels are placed using the *Curved* placement option for label candidate generation, so that they follow the shape of the roads. They are set to be placed *On line* features, and those line features are not themselves rendered. The *Priority* parameter is set to 6 on a 0-10 scale; *Always Show* is set to *True* for displaying every label for the layer. Label repetition (*Repeat*) is applied on the road labels, so that longer roads can be labelled with more than one label. The repetition is set to 50 millimeters. The road text labels have the font Univers 57 Condensed. They are rendered white, and their size is defined by adding a field to the attribute table of the road layer, where the label size is set for each road feature in the layer. Labels for major roads are set to be 13pt in size. Other road labels have a size of 8pt.

Landmark building text labels are rendered after the road labels because their placement is permitted to be changed and modified quite much, as stated in section 5.1.3. Therefore, the *Priority* parameter is set to 2, i.e. lower than for the road labels. The label candidate placement is set to *Around centroid* since this allows multi-line labels to be aligned depending on placement. Longer labels are stacked by applying the #-character to wrap the label text.

The character is inserted in attribute names for determining where a new line begins. The alignment of the landmark building text labels is set to follow the placement of them in relation to their features. This means that labels placed on the left side of their polygon features are right-aligned, and vice versa. Labels placed close to the center of their polygons are centered. As with the road labels, *Always Show* is set to *True*. The landmark building labels are rendered white, with the font NJFont Bold and a size of 12pt.

Village name labels are rendered lastly by the label placement program because they are not meant to be covered by any other labels, and because they may overlap other map information. The village labels belong to a polygon layer whose features are not rendered. Label candidates are generated using the *Offset from centroid* placement option. The default *Quadrant* placement is applied, i.e. *Over* the polygon centroid point. The village labels are set to *Always Show*. They are rendered with a 55pt size, in the font NJFont Light; they are white and have an opacity of 80%.

Code from Wei (2020) is added to the label placement program to produce the simplification images described in section 5.2. The default *Automated Placement Engine* settings are used in the production of that map: For line features, the number of label candidates per cm is set to 5. For polygon features, the number is set to 2.5 candidates per cm<sup>2</sup>. Furthermore, the default search method *chain* is utilized. Village labels are not included in the rasterized images, as they would disturb the evaluation of the other labels.

There was an effort to set the global label placement parameters by using the API. However, this did not succeed: The API for the global parameters could not be reached properly, and it is unclear as to why. Therefore, global label placement parameters are manually set using the *Automated Placement Engine* in QGIS. Four maps are produced, for which the number of candidates for line and polygon features are set at different values. These values are presented in *table 5.1*. The optimization method used is the *chain* method, since it is the default setting, and because other optimization methods could not be set. For all four maps, the layer specific label parameters and the symbol label placement described above are left unchanged.

*Table 5.1: The number of candidates per cm and cm<sup>2</sup> generated for line and polygon features, respectively, applied for each of the four maps produced.*

Map	Line: No. of candidates (per cm)	Polygon: No. of candidates (per cm <sup>2</sup> )
1	5	2.5
2	10	5
3	50	25
4	80	40

Due to the limited time frame of the study, the labels placed need to be prioritized. This results in some label types not being rendered, and therefore not being included in the practical study. For example, the cycle hire, bus stop, and the larger station entrance symbol label are not rendered. These are not as prioritized due to their placement trait: they do not have a set point location like for example taxi rank and car park symbols have. Moreover, the neighborhood and River Thames text labels are not included on the maps, neither are the retail unit icons.

## 6. Results

### 6.1 Evaluation according to T-Kartor's labelling rules

This section presents the label placement results and to what degree they satisfy the label placement rules defined by T-Kartor, found in section 5.1. The effect of increasing the number of labels candidates generated for line and polygon features is also reviewed. The four maps produced are externally appended as related material. This evaluation is not related to the quality evaluation done by Wei (2020). The overall legibility of the produced maps is acceptable, though there are places where the labels are difficult to read, mostly due to label-label overlaps. There is a level of ambiguity in some landmark building labelling, where two building features lie close to each other and the feature-label association is quite weak or confusing. The label placement follows the desired rendering order: symbol labels are rendered before text labels.

All the symbol labels that are labelled are placed on their point location, since they are not allowed to be placed anywhere else (without a leader line). Some labels overlap landmark building features, which is permitted. Even though the road and landmark building labels are blocked from overlapping symbol labels, overlapping occurs at some places. However, the blocking reduced the number of overlaps. The symbol-symbol overlaps, although few, are not solved. The symbol label placement generally is of a good quality. In *figure 6.1*, symbol label placement is illustrated.





Figure 6.1: Symbol labelling from Map 3. Three types of symbol labels are present. Based on data from © T-Kartor Group AB.

The general road label placement rules are met: The labels are placed on their roads, and they follow the shape of their roads since they are curved. The curvature of the labels is limited to retain legibility. However, many labels appear straight because their roads are straight, as shown in figure 6.2.

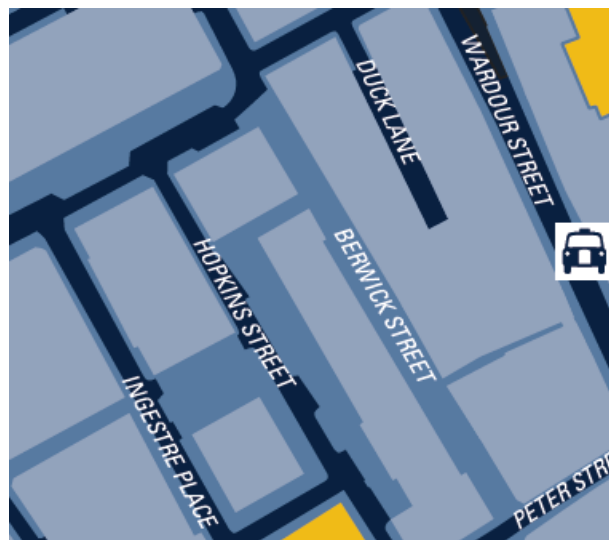


Figure 6.2: Road labelling from Map 4. The roads and their labels are straight. Based on data from © T-Kartor Group AB.

The worst quality of the road labelling is that some labels are suppressed. According to the rules, no label should be suppressed. Multi-line labelling and shortened label text is not implemented: this could be needed to better fit labels within roads. The repetition set for road labels only affects a small proportion of the road labels, and the label repetition produced is not good: some road features have too many labels, while others have too few label repetitions, with long road segments remaining unlabeled. Label alignment of parallel roads

seem to depend on chance: they are far from always placed uniformly. Many road labels follow the rule that they should not be placed at intersections or at the end of roads: they should be centered on the road feature. Still, a large proportion of them do not follow this rule, as shown in *figure 6.3*. When space is limited the rule can be difficult to follow, as seen in e.g. *figure 5.3*. Many symbol and landmark building labels are placed so that they overlap roads, which is hard to avoid.



*Figure 6.3: Road labeling from Map 3. Many road labels tend to be placed close to intersections or end of roads. Based on data from © T-Kartor Group AB.*

Landmark building labels are placed horizontally, according to the placement rule. Very few of them are placed entirely within their polygons. However, almost all the labels are partially within their polygons, which can be seen in *figure 6.4*. The stacking of label texts gives positive results: A larger part of labels is placed within their landmark building polygons. The alignment of stacked landmark building labels quite successfully follows the rule that labels should be aligned depending on their position. For many labels it is the aligned part that is inside the polygon, which is desirable. The label alignment of landmark building labels is exemplified in *figure 6.4*. The label color is always set to white: This does not follow the rule that only labels overlapping roads should be white.



Figure 6.4: Landmark building labelling from Map 4. The aligned part of a label is often the part that is inside the polygon. The label alignment is dependent on the placement: It can be left aligned, right aligned, or centered. Based on data from © T-Kartor Group AB.

The village labels are placed at or close to the centroid of their polygons, which is wanted. However, this cannot be seen from these maps since the village polygon features are not rendered. Village can overlap other map objects, which they do. A village labels can be seen in figure 6.5.

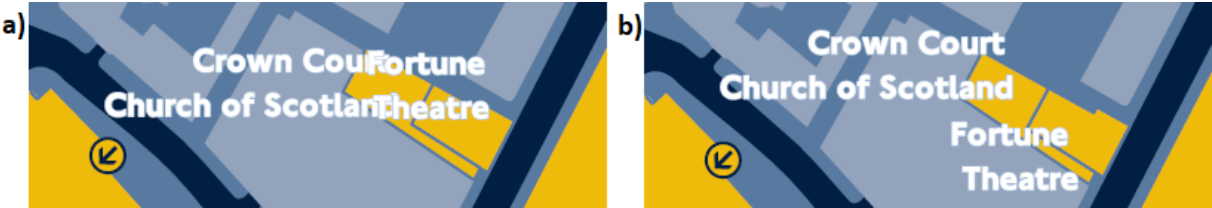


Figure 6.5: A village label from Map 1. It overlaps other map information. Based on data from © T-Kartor Group AB.

The four maps produced, with varying numbers of candidates generated, give some different label placement results, but they do not vary by much. Map 1, which labelling utilized the fewest candidates, has the most deviant labelling. Map 2, 3, and 4 have more similar label placement. Since symbol labels are rendered as features, they are placed at the same position on all four maps.

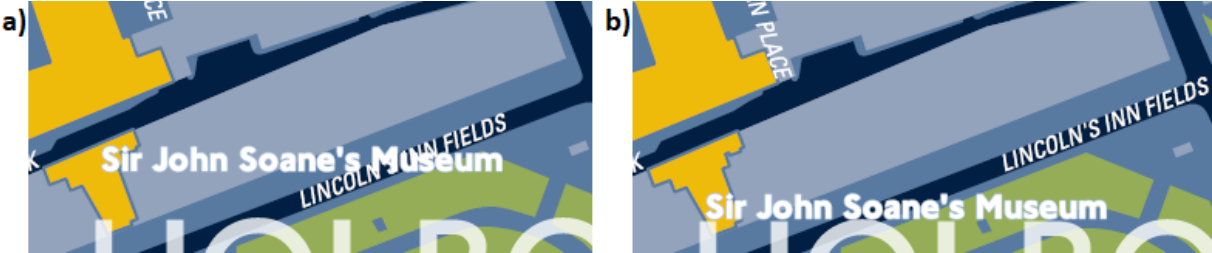
An increase in the number of candidates generated per cm for line feature labels does not increase the quality of the road label placement by much. Many road labels are placed at the same location on all four maps. Most of the labels that do have different positions on the maps only have a placement difference of a few millimeters.

The label placement of landmark buildings is not greatly affected by adding more label candidates per area unit. The majority of the labels have the same placement on every map. Some overlaps between landmark building labels are resolved with a higher number of candidates. An example of this is found in *figure 6.6*. However, other such overlaps are the same on all maps. The village labels are placed the same on each of the four maps.



*Figure 6.6: Comparison of text label placement between a) Map 1 and b) Map 2. Map 1 contains an overlap between two landmark building labels. In Map 2 these labels do not overlap. Map 1 has 2.5 candidates per cm<sup>2</sup>, and Map 2 has 5 candidates per cm<sup>2</sup>. Based on data from © T-Kartor Group AB.*

Some label overlapping is avoided by increasing the candidate numbers, e.g. overlaps between road and landmark building labels, as seen in *figure 6.7*. The overlap only occurs in *Map 1*. In some overlap cases, overlapping proportion increased with increased number of candidates. Other label overlapping remained the same on all four maps, being unaffected by increasing the number of candidates. This is illustrated by *figure 6.8*.



*Figure 6.7: Comparison of text label placement between a) Map 1 and b) Map 4. Map 1 contains an overlap; Map 4 does not. Map 1 has 2.5 candidates per cm<sup>2</sup>, and Map 4 has 40 candidates per cm<sup>2</sup>. Based on data from © T-Kartor Group AB.*



Figure 6.8: Overlaps between road and landmark building labels, on Maps 1, 2, 3, and 4. Based on data from © T-Kartor Group AB.

## 7. Discussion

### 7.1 Interpretation of results

This thesis aims at designing a label placement program for static city maps, to decrease the amount of time spent on manual label placement. The labelling produced is meant to follow the company T-Kartor's label placement rules, which are obtained through an interview. The labelling produced partially follows the label placement rules, but the overall label placement quality is nevertheless poor to moderate. A substantial proportion of the labels in all four maps need manual editing for better placement.

In this study, symbols are rendered as features. Still, they could in theory also be rendered as labels, but an efficient way of implementing that in the QGIS environment has not been found in this study. The symbol labels are relatively easy to label because they are to be placed at specific point locations. Leader lines can be added to symbols in the post-processing if labels cluster and the only solution is to move symbols. The symbols do not have a problem with automatic suppression since they are rendered as features. Therefore, all labelled symbols are displayed on the maps. Much of the text-symbol overlapping is avoided by applying label blocking for the symbol labels. However, symbol-symbol overlaps are not solved through this method.

There are quite many problems with the road labelling. Firstly, some of the road labels are not displayed. All line feature labels of a layer can only be displayed if their placement is set to *Horizontal*. Selecting *Parallel* or *Curved* candidate placement results in labels being automatically suppressed, due to placement issues such as label overlapping or fitting. Some road features may be shorter than their labels, but even in such cases, at least one label candidate position should be created (Ertz et al. 2009). This suppression problem has

previously been reported as a software bug in the QGIS community. The fact that all road labels cannot be displayed if they are curved or parallel is a significant disadvantage because these are two important labelling rules to be followed: that no label should be suppressed, and that road labels should follow the shape of roads (stated in sections 5.1.5 and 5.1.2 respectively). This means that the rule stating that all labels should be shown cannot be followed in the labelling. The *Horizontal* placement should not be applied since that would result in unsatisfactory road labelling. Apart from the suppression problem, a good label repetition is difficult to set for road labels. Furthermore, curved labels do not remain curved when editing them manually. This is bad since many curved labels need to be manually moved in the post-processing.

The landmark building labels can all be shown. This is because their candidate placement is set to *Around centroid*. Their overall placement may be a little better than that of the road labels. However, many that are partially placed outside of their polygon could have fitted inside. The placement dependent alignment mostly yielded good results. A more extensive use of label stacking for the landmark buildings could improve their placement, by having a larger proportion of the labels inside the polygons. This could also potentially decrease overlapping that longer landmark building labels cause, e.g. as in *figures 6.7* and *6.8*. Setting the *Priority* parameter at different values for text labels did not affect the label placement whatsoever. This parameter is meant to control what labels that are labelled in overlap conflicts.

It is not surprising that the label placement quality is not increased dramatically by increasing the number of generated candidates for line and polygon feature labels, while only applying one search method. As stated previously, the *chain* search method that is used is relatively simple and can only output labelling result up to a certain quality. Therefore, an increase in the number of candidates can allow for a somewhat better placement, but the labelling is still limited by the search method applied. In section 7.3 the utilization of other search methods is discussed.

Some labels types, e.g. retail unit icons and cycle hire symbols, were not implemented in the label placement program, and therefore were not labelled. It is difficult to assess how this affected the label placement results, but as more labels are placed, the harder it is to avoid overlaps. More label types would need to be labelled to make a fairer analysis of the label placement results, and to better know if the label placement program could be of practical use

for T-Kartor. Lastly, the labelling results would need to be quantified using more robust methods, before a more scientifically trusting analysis of the label placement can be made.

## 7.2 Algorithm suitability

Since the PAL labelling library is based on combinatorial optimization algorithms, it uses heuristics to find optimal labelling solutions. The practical study of this thesis is heavily limited in terms of the optimization methods that can be used: Only PAL's *chain* method is applied in the labelling. This is the simplest and fastest of the search methods available in PAL (Ertz et al. 2009). It is quite likely that by implementing other PAL search methods, the labelling would be of a higher quality. For example, POPMUSIC search methods can produce better label placement because smaller parts, or sub-problems, are improved separately from the rest of the solution problem (Laurent et al. 2009), which allows for a more flexible optimization process. In previous studies, POPMUSIC search methods have found label placement solutions of relatively high quality, compared to the *chain* method (Ertz et al. 2009; Laurent et al. 2009). Applying the POPMUSIC search methods in PAL (*pop\_chain* and *pop\_tabu\_chain*) could therefore result in better label placement.

Sure, these more complex optimization methods demand more computational time (Ertz et al. 2009). But this is of lesser concern, if the computation does not take an excessive amount of time. One of the most needed qualities for the optimization method used is that it should allow all labels to be placed, since this one of T-Kartor's map labelling rules. It is uncertain if the methods *pop\_chain* and *pop\_tabu\_chain* possess this quality. However, according to Ertz et al. (2009), the search method *pop\_tabu* produces labelling where all labels are placed. This method is available in PyQGIS (QGIS Development Team 2020a). As more labels are shown, label overlap often is increased. However, this is of lesser concern, as it can be edited in the post-processing. Another needed feature is that line feature labels should be able to be curved, and have their curvature edited. This is not possible in QGIS yet, as curved labels are automatically made straight when manually moving them.

According to QGIS Development Team (2019), using a larger number of candidate label positions for line and polygon labels generally increases the label placement quality. But using very large numbers results in slower labelling performance. Thus, one should try to apply quite many candidate positions for each feature, while avoiding slowing down the labeling speed too much. However, in this practical study, the labelling quality did not

increase much by having more candidates generated for line and polygon features, and the difference in computational time for the four maps was not noticeable.

Using other combinatorial optimization algorithms which apply other optimization methods could produce label solutions of different qualities. How the objective function is constructed is of high relevance, since it determines the labelling preferred by the algorithm. For the city wayfinding maps produced by T-Kartor, two of the most important criteria for an objective function are (1) placement quality, i.e. label-feature association, and (2) overlapping. The placement quality criteria could be hard to define, but it could be based on prioritization of candidate placement. The overlap criteria could number the label-label overlaps of a solution. Furthermore, the algorithm must not under any condition remove labels. This method would result in a labelling solution that minimizes overlapping, while also placing all the labels in the most optimal way.

Applying a simulated annealing optimization method in this practical study would have been interesting, since it can produce labelling of relatively high quality (Christensen et al. 1995; Edmondson et al. 1997; Wagner and Wolff 1998; Zhang and Harrie 2006a; Ertz et al. 2009). The use of the temperature parameter, which makes it possible to escape local minima of the cost function, makes the algorithm flexible and different from the optimization methods found in PAL. The chain search methods in PAL do not use a temperature. However, like with the simulated annealing method, the *pop\_tabu\_chain* method can also accept changes that result in decreased label placement quality. This property may result in new labelling solutions that would not have been found by other methods. Therefore, it would be interesting to quantitatively compare the label placement quality produced by the optimization methods *chain*, *pop\_chain*, *pop\_tabu*, *pop\_tabu\_chain*, and simulated annealing.

There are other algorithms that probably not would produce good label placement. For example, greedy algorithms are often too simple to find labelling solutions of high quality. Previous studies have found that this algorithm type is not the most suitable for finding good solutions to the label placement problem (Christensen et al. 1995; Wagner and Wolff 1998). More complex algorithms are needed to produce good labelling.

Apart from combinatorial optimization algorithms, other kinds of label placement algorithms could be tested, e.g. sequential algorithms or slider models. The sequential method reduces the labelling options of many features because it is a depth-first method. This means that the labelling options become increasingly limited as more features are labelled, as candidate



positions for labels not yet placed are deleted if they overlap placed labels (Cook and Jones 1990). This could imply that algorithms applying sequential label placement methods would not produce good label placement solutions for city wayfinding maps. If PAL had a slider method of placing labels, the labelling of line and polygon features such as roads and landmark buildings may have been of a higher quality, with fewer overlaps and better placement. This is because slider models enable a higher number of available label positions without largely increasing the running time of the algorithm (van Kreveld et al. 1999).

### 7.3 Automated and manual label placement

High quality label placement can be difficult to achieve by solely relying on automated placement. It is probably easier to obtain good label placement by combining the efficiency of automated labelling together with manual label editing, which skilled cartographers are good at. This combination can be applied in different ways, e.g. by integrating manual editing in an iterative process as done by Klute et al. (2019), or more commonly, by first running a label placement algorithm and then manually editing labels to achieve a final labelling. T-Kartor relies on manual label editing, and they will likely continue to do so, whether they will switch from using Maplex to another label placement program.

### 7.4 Future research

First and foremost, similar future studies could apply and compare all search methods available in PAL. This could result in quite different labelling, and it could be the simplest method of improving label placement quality in QGIS. In this study, only the *chain* method was applied. There is a large probability that some of the other search methods would produce higher quality labelling, at least for the text labels. However, the symbols and icons would most likely still be best labelled as symbol features and would thus not be affected by using other optimization methods. Layer specific labelling parameters should also be explored further to find optimal parameter settings. To accurately evaluate and compare labelling results, the label placement needs to be quantified. This could be done by for example counting the number of overlaps present on each map, or by using a quality level schema to grade the label placement, as done by Wei (2020). The labelling results of the practical study are not good enough for T-Kartor to apply the label placement program in their map production, but it could be a starting point that leads to a more robust method. More research

would be needed to further investigate the possibilities of finding a better automated label placement method.

## 8. Conclusions

This thesis investigates in the possibilities of utilizing other label placement software for the company T-Kartor's map production. Label placement rules is retrieved from an interview with T-Kartor. Some general label placement rules prove to be similar or the same as is found in the literature. For example, label overlapping and ambiguity in the label-feature association should be minimized when labelling. However, other labelling rules differ from the literature, e.g. that no label may be automatically removed in the labelling of T-Kartor's maps. This study follows the rules set up by T-Kartor. The choice of software and algorithm applied in the practical study is based on a literature review of methods for automated label placement. The open-source applications QGIS and PAL are chosen. PAL uses combinatorial optimization methods for point, line, and polygon feature label placement. PAL algorithm parameters can be accessed and set both with the QGIS GUI and the QGIS API.

The label placement results are evaluated manually. The program generates labelling of poor to moderate quality, where many labels would need manual editing to be better placed. Therefore, the label placement program developed in this study is not of a quality that allows T-Kartor to use it professionally. The global parameter settings need to be explored further, but also the layer specific parameters. Furthermore, the label placement needs to be quantified in an evaluation. The results of the practical study should be viewed as a starting point of a more extensive research project on finding a better method for automated label placement, in which Wei's work (2020) also contribute.

## References

- Christensen, J., J. Marks, and S. Shieber. 1995. An empirical study of algorithms for point-feature label placement. *ACM Transactions on Graphics*, 14: 203-232.
- Cook, A. C., and C. B. Jones. 1990. A Prolog Rule-Based System for Cartographic Name Placement. In *Computer Graphics Forum*, 109-126.
- Downey, A. B. 2012. *Think Python: How to Think Like a Computer Scientist, Version 2.0.17*. Needham, Massachusetts: Green Tea Press.
- Edmondson, S., J. Christensen, J. Marks, and S. Shieber. 1997. A general cartographic labeling algorithm. *Cartographica: The International Journal for Geographic Information*, 33: 13-24.
- Ertz, O., M. Laurent, D. Rappo, A. Sae-Tang, and E. Taillard. 2009. PAL-A cartographic labelling library. *Position IT July 2009*: 56-61.
- Imhof, E. 1975. Positioning names on maps. *The American Cartographer*, 2: 128-144.
- Jones, C. B. 1989. Cartographic name placement with Prolog. *IEEE Computer Graphics and Applications*, 9: 36-47. DOI: 10.1109/38.35536
- Klute, F., G. Li, R. Löffler, M. Nöllenburg, and M. Schmidt. 2019. Exploring Semi-Automatic Map Labeling. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 13-22.
- Kraak, M.-J., and F. Ormeling. 1996. *Cartography: Visualization of Spatial Data*. Harlow: Addison Wesley Longman Limited.
- Laurent, M., É. D. Taillard, O. Ertz, F. Grin, D. Rappo, and S. Roh. 2009. From point feature label placement to map labelling. In *Proceedings, metaheuristic international conference (MIC'09), Hamburg*. Citeseer.
- Longley, P. A., M. F. Goodchild, D. J. Maguire, and D. W. Rhind. 2011. *Geographic information systems and science*. John Wiley & Sons.
- Marín, A., and M. Pelegrín. 2018. Towards unambiguous map labeling - Integer programming approach and heuristic algorithm. *Expert Systems with Applications*, 98: 221-241.
- Oliphant, T. E. 2007. Python for scientific computing. *Computing in Science & Engineering*, 9: 10-20.
- QGIS Development Team. 2019. QGIS testing. *QGIS User Guide: Release testing*.
- QGIS Development Team. 2020a. QGIS 3.12 Geographic Information System API Documentation. Retrieved from <https://qgis.org/api/3.12/>
- QGIS Development Team. 2020b. QGIS 3.12 geographic information system. *Open source geospatial Foundation project*.
- Robinson, A. H., R. D. Sale, J. L. Morrison, and P. C. Muehrcke. 1984. *Elements of Cartography*. New York, Chichester, Brisbane, Toronto, and Singapore: John Wiley & Sons.
- Russell, S. J., and P. Norvig. 1995. *Artificial intelligence: A modern approach*. Englewood Cliffs, New Jersey: Prentice Hall.
- Rylov, M., and A. Reimer. 2017. A practical algorithm for the external annotation of area features. *The Cartographic Journal*, 54: 61-76.
- Sherman, G. E., T. Sutton, R. Blazek, and L. Luthman. 2004. *Quantum GIS User Guide*.

- Slocum, T. A., R. B. McMaster, F. C. Kessler, and H. H. Howard. 2005. *Thematic Cartography and Geographic Visualization*. Pearson Prentice Hall.
- Steiniger, S., and A. J. S. Hunter. 2013. The 2012 free and open source GIS software map – A guide to facilitate research, development, and adoption. *Computers, environment and urban systems*, 39: 136-150.
- Strijk, T., and M. van Kreveld. 2002. Practical extensions of point labeling in the slider model. *GeoInformatica*, 6: 181-197.
- Taillard, É. D., and S. Voss. 2002. POPMUSIC - Partial optimization metaheuristic under special intensification conditions. In *Essays and surveys in metaheuristics*, 613-629 pp.: Springer.
- Transport for London. 2009. Street map design standard - Issue 1.
- Transport for London. 2011. Street map design standard - Issue 2.
- Wagner, F., and A. Wolff. 1998. A combinatorial framework for map labeling. In *International Symposium on Graph Drawing*, 316-331. Springer.
- van Dijk, S. 2001. *Genetic algorithms for map labeling*. Utrecht University, Netherlands.
- van Dijk, S., D. Thierens, and M. De Berg. 2002. Using genetic algorithms for solving hard problems in GIS. *GeoInformatica*, 6: 381-413.
- van Kreveld, M., T. Strijk, and A. Wolff. 1999. Point labeling with sliding labels. *Computational Geometry*, 13: 21-47.
- Wei, L. 2020. An artificial intelligence method for text labelling. Unpublished.
- Wolff, A. 2002. Automated label placement in theory and practice. PhD Thesis.
- World Wide Web Consortium. 2018. Scalable Vector Graphics (SVG) 2. Retrieved 17 June 2020, from <https://www.w3.org/TR/SVG2/>.
- Yoeli, P. 1972. The logic of automated map lettering. *The Cartographic Journal*, 9: 99-108.
- Zhang, Q., and L. Harrie. 2006. Placing text and icon labels simultaneously: A real-time method. *Cartography and Geographic Information Science*, 33: 53-64.
- Zhang, Q.-n., and L. Harrie. 2006. Real-time map labelling for mobile applications. *Computers, environment and urban systems*, 30: 773-783.
- Zoraster, S. 1997. Practical results using simulated annealing for point feature label placement. *Cartography and Geographic Information Systems*, 24: 228-238

## Appendix A: Program code

```
### Python script for the Code Editor in QGIS ###
#####

## Name: LabelPlacementProgram_Cederholm2020.py
## Created: 19 March 2020
## Author: Pontus Cederholm

# Importing the os module to enable functionality related to the
operating system.
import os

# Loading the London project.
project = QgsProject.instance()
project.read("C:/Users/Admin/Desktop/Qgis_Cartography_London/Qgis_Lon
don/London.qgz")

# Setting the scale.
canvas = qgis.utils.iface.mapCanvas()
canvas.zoomScale(2250)

### Symbol label placement ###

# Placing the taxi symbol.
# Defining the taxi layer. Name: 'LMF Taxi ranks'.
taxi = project.mapLayersByName('LMF Taxi ranks')[0]
# Setting it as the active layer.
iface.setActiveLayer(taxi)
# Setting the taxi SVG.
taxiIcon =
QgsSvgMarkerSymbolLayer("C:/OSGeo4W64/apps/qgis/svg/TfL_Other/IconTax
i.svg")
# Setting blocking.
taxi.loadNamedStyle('C:/Users/Admin/Desktop/BlockTaxi.qml')
taxiIcon.setSize(6.5) # Millimeters is the default unit.
taxi.renderer().symbol().changeSymbolLayer(0, taxiIcon)
taxi.triggerRepaint()

# Placing the car park symbol.
carPark = project.mapLayersByName('LMF Car Parks')[0]
iface.setActiveLayer(carPark)
carParkIcon =
QgsSvgMarkerSymbolLayer("C:/OSGeo4W64/apps/qgis/svg/TfL_Other/CarPark
.svg")
carParkIcon.setSize(6.5)
carPark.loadNamedStyle('C:/Users/Admin/Desktop/BlockCarParks.qml')
carPark.renderer().symbol().changeSymbolLayer(0, carParkIcon)
carPark.triggerRepaint()

# Placing the police station symbol.
policeSt = project.mapLayersByName('LMF Police Stations')[0]
iface.setActiveLayer(policeSt)
policeStIcon =
QgsSvgMarkerSymbolLayer("C:/OSGeo4W64/apps/qgis/svg/TfL_Other/Police.
svg")
policeStIcon.setSize(10)
policeSt.loadNamedStyle('C:/Users/Admin/Desktop/BlockPoliceSt.qml')
policeSt.renderer().symbol().changeSymbolLayer(0, policeStIcon)
```

```

policeSt.triggerRepaint()

# Placing the post office symbol.
post = project.mapLayersByName('LMF Post Offices')[0]
iface.setActiveLayer(post)
postIcon =
QgsSvgMarkerSymbolLayer("C:/OSGeo4W64/apps/qgis/svg/TfL_Other/IconPos
t.svg")
postIcon.setSize(10)
post.loadNamedStyle('C:/Users/Admin/Desktop/BlockPost.qml')
post.renderer().symbol().changeSymbolLayer(0, postIcon)
post.triggerRepaint()

# Placing the ticket agent symbol.
ticket = project.mapLayersByName('LMF Ticket Agents')[0]
iface.setActiveLayer(ticket)
ticketSymb =
QgsSvgMarkerSymbolLayer("C:/OSGeo4W64/apps/qgis/svg/TfL_Other/TicketA
gents.svg")
# The symbol should be 10mm in height. Here the width is defined so
that the height is 10mm.
ticketSymb.setSize(17.333356)
ticket.loadNamedStyle('C:/Users/Admin/Desktop/BlockTicket.qml')
ticket.renderer().symbol().changeSymbolLayer(0, ticketSymb)
ticket.triggerRepaint()

# Placing the information center symbol.
info = project.mapLayersByName('LMF Info Centres')[0]
iface.setActiveLayer(info)
# Setting the TrueType font: font family, character, size (mm), and
color.
# The symbol size should be 6.5mm, but 13mm is needed to achieve the
correct size.
infoSymb = QgsFontMarkerSymbolLayer("T-K TfLsymb06", "๕๓", 13,
QColor(255, 255, 255, 255))
info.loadNamedStyle('C:/Users/Admin/Desktop/BlockInfo.qml')
info.renderer().symbol().changeSymbolLayer(0, infoSymb)
info.triggerRepaint()

# Placing the building entrance symbol.
bEntrance = project.mapLayersByName('LMF Building Entrances')[0]
iface.setActiveLayer(bEntrance)
bEntranceSymb =
QgsSvgMarkerSymbolLayer("C:/OSGeo4W64/apps/qgis/svg/TfL_Other/Buildin
gEntrance.svg")
bEntrance.loadNamedStyle('C:/Users/Admin/Desktop/BlockBEntrance.qml')
# Using the rotation attribute field to set the rotation of the
symbols.
bEntranceSymb.dataDefinedProperties().setProperty(QgsSymbolLayer.Prop
ertyAngle, QgsProperty.fromField('ROTATION'))
bEntranceSymb.setSize(4.6)
bEntrance.renderer().symbol().changeSymbolLayer(0, bEntranceSymb)
bEntrance.triggerRepaint()

# Placing the station entrance symbol.
stEntrance = project.mapLayersByName('LMF Station Entrance')[0]
iface.setActiveLayer(stEntrance)
stEntranceSymb =
QgsSvgMarkerSymbolLayer("C:/OSGeo4W64/apps/qgis/svg/TfL_Station/DarkI
nnerStationUG.svg")
stEntranceSymb.setSize(5)

```

```

stEntrance.loadNamedStyle('C:/Users/Admin/Desktop/BlockStation.qml')
stEntrance.renderer().symbol().changeSymbolLayer(0, stEntranceSymb)
stEntrance.triggerRepaint()

### Text label placement ###

# Defining the road layer.
roads = project.mapLayersByName('LMF Road_lines')[0]
iface.setActiveLayer(roads)
# Removing the road line symbol.
roads.loadNamedStyle('C:/Users/Admin/Desktop/NoSymbolRoads.qml')
# Labelling roads.
roadsSettings = QgsPalLayerSettings()
roadsTxtFormat = QgsTextFormat()
roadsTxtFormat.setFont(QFont("Univers LT Std 57 Cn"))
roadsTxtFormat.setColor(QColor.fromCmyk(0, 0, 0, 0)) # White
roadsSettings.setFormat(roadsTxtFormat)
roadsSettings.fieldName = "ROADNAME"
# Setting candidate label placement.
roadsSettings.placement = 3 # Parallel: 2; Curved: 3.
roadsSettings.placementFlags = roadsSettings.OnLine
# Defining label size. Referring to an attribute field with size
values.
pcRoads = QgsPropertyCollection('pcRoads')
roadLabelSize = QgsProperty()
roadLabelSize.setField("LabelSize")
pcRoads.setProperty(0, roadLabelSize) # 0 = Size property.
# Setting label priority.
roadsPrio = QgsProperty()
roadsPrio.setStaticValue(10)
pcRoads.setProperty(87, roadsPrio)
# Set to show all labels. Does not work for curved and parallel
labels.
roadsShow = QgsProperty()
roadsShow.setStaticValue(1) # 1 = True.
pcRoads.setProperty(20, roadsShow)
roadsSettings.setDataDefinedProperties(pcRoads)
# Setting the maximum angle between curved characters.
roadsSettings.maxCurvedCharAngleIn = 30
roadsSettings.maxCurvedCharAngleOut = 40
roadsSettings.repeatDistance = 50 # Label repetition (mm).
# Enabling the labelling.
roadsSettings.enabled = True
roadsSettings = QgsVectorLayerSimpleLabeling(roadsSettings)
roads.setLabelsEnabled(True)
roads.setLabeling(roadsSettings)
roads.triggerRepaint()

# Labelling landmark bulidings.
buildings = project.mapLayersByName('LMF Landmark Building')[0]
iface.setActiveLayer(buildings)
buildingsSettings = QgsPalLayerSettings()
buildingsTxtFormat = QgsTextFormat()
buildingsTxtFormat.setFont(QFont("NJFont Bold"))
buildingsTxtFormat.setColor(QColor.fromCmyk(0, 0, 0, 0)) # White
buildingsTxtFormat.setSize(12) # 12pt
# Setting the stroke.
buildStroke = QgsTextBufferSettings()
buildStroke.setEnabled(True)
buildStroke.setSize(0.75)
buildStroke.setSizeUnit(4) # Unit: Points.

```

```

buildStroke.setColor(QColor.fromCmyk(53, 27, 0, 31))
buildingsTxtFormat.setBuffer(buildStroke)
buildingsSettings.setFormat(buildingsTxtFormat)
buildingsSettings.fieldName = "Name_stacked"
# Setting candidate label placement.
buildingsSettings.placement = 0 # Around centroid = 0
pcBuild = QgsPropertyCollection('pcBuild')
buildLabelStack = QgsProperty()
buildLabelStack.setStaticValue("#") # Stacking labels using the #-
character.
pcBuild.setProperty(31, buildLabelStack)
# Setting the alignment.
# Labels are aligned depending on their placement.
buildingsSettings.MultiLineAlign = 3
# Setting label priority.
buildingPrio = QgsProperty()
buildingPrio.setStaticValue(2)
pcBuild.setProperty(87, buildingPrio)
# Set to show all labels.
buildingsShow = QgsProperty()
buildingsShow.setStaticValue(1)
pcBuild.setProperty(20, buildingsShow)
buildingsSettings.setDataDefinedProperties(pcBuild)
# Enabling the labelling.
buildingsSettings.enabled = True
buildingsSettings = QgsVectorLayerSimpleLabeling(buildingsSettings)
buildings.setLabelsEnabled(True)
buildings.setLabeling(buildingsSettings)
buildings.triggerRepaint()

# Labelling villages.
village = project.mapLayersByName('LMF Villages')[0]
iface.setActiveLayer(village)
villageSettings = QgsPalLayerSettings()
villageTxtFormat = QgsTextFormat()
villageTxtFormat.setFont(QFont("NJFont Light"))
villageTxtFormat.setColor(QColor.fromCmyk(0, 0, 0, 0)) # White
villageTxtFormat.setSize(55) # 12pt
villageTxtFormat.setOpacity(0.80)
villageSettings.setFormat(villageTxtFormat)
villageSettings.fieldName = "NAME2"
# Setting candidate label placement.
villageSettings.placement = 1 # Offset from centroid = 1
pcVillage = QgsPropertyCollection('pcVillage')
# Setting label priority.
villagePrio = QgsProperty()
villagePrio.setStaticValue(1)
pcVillage.setProperty(87, villagePrio)
# Set to show all labels.
villageShow = QgsProperty()
villageShow.setStaticValue(1)
pcVillage.setProperty(20, villageShow)
villageSettings.setDataDefinedProperties(pcVillage)
# Enabling the labelling.
villageSettings.enabled = True
villageSettings = QgsVectorLayerSimpleLabeling(villageSettings)
village.setLabelsEnabled(True)
village.setLabeling(villageSettings)
village.triggerRepaint()

# Set visible layers.

```



```

root = project.layerTreeRoot ()
root.children ()
ids = root.findLayerIds ()
# Setting area layers to be visible.
AREAS = root.findGroup ("AREAS")
AREAS.setItemVisibilityChecked (True)
# Setting the layers with labels to be visible.
roadL = root.findLayer (roads.id ())
roadL.setItemVisibilityChecked (True)
villageL = root.findLayer (village.id ())
villageL.setItemVisibilityChecked (True)
carParkL = root.findLayer (carPark.id ())
carParkL.setItemVisibilityChecked (True)
bEntranceL = root.findLayer (bEntrance.id ())
bEntranceL.setItemVisibilityChecked (True)
ticketL = root.findLayer (ticket.id ())
ticketL.setItemVisibilityChecked (True)
taxiL = root.findLayer (taxi.id ())
taxiL.setItemVisibilityChecked (True)
stEntranceL = root.findLayer (stEntrance.id ())
stEntranceL.setItemVisibilityChecked (True)
postL = root.findLayer (post.id ())
postL.setItemVisibilityChecked (True)
policeL = root.findLayer (policeSt.id ())
policeL.setItemVisibilityChecked (True)
infoL = root.findLayer (info.id ())
infoL.setItemVisibilityChecked (True)

##### End of program #####

```

## Appendix B: Interview

Date: 25<sup>th</sup> February 2020.

Place: T-Kartor's head office in Kristianstad, Sweden

Interviewer: Pontus Cederholm – Thesis author

Interviewee: Åsa Nilsson – Project administrator at T-Kartor

### **Generellt sätt, vilka är de främsta kriterierna som ni använder för god text- och symbolsättning?**

Läsbarhet är det viktigaste. Att man förstår vilka objekt som texterna hör till, så att det inte uppstår tvetydighet. Detta för att man ska kunna hitta vägar och platser via kartan.

### **Försöker ni alltid placera texter på samma vis i relation till deras kartobjekt? T.ex. att text alltid helst ska placeras över ett punktobjekt, alltid längs med och på ett linjeobjekt, och alltid inuti en polygon?**

En linjetext ska aldrig sluta i en korsning, för då vet man inte om den vägen fortsätter efter korsningen, eller om en annan tar vid. Därför ska texten gå över korsningen. För textsättning till punkter så är prioriteringsordningen: (1) Uppe till vänster om punkten; (2) Uppe till höger om punkten; (3) Nere till vänster om punkten; (4) Nere till höger om punkten. Denna regel gäller dock mest för symboler och ikoner, och är inte speciellt relevant för texter, eftersom de flesta texterna hör till vägar, gator, och byggnader, vilka inte är punktobjekt. Det är mest ikoner och symboler som representerar punkter, snarare än texter.

Symboler kan sättas över både punkter och polygoner. Parkeringsymbolen hör till en yta, därför sätts den med en polygon. Medan t.ex. taxisymbolen placeras på en punkt. Prioriteringsordningen för punkter kan också appliceras på polygoner och deras centerpunkter för att sätta texter för polygoner, t.ex. byggnader.

**Angående textplacering för punkter, hur prioriterar ni placeringen i relation till punkten, i de fall där de inte placeras rakt på objektets plats (dvs punkten är ej inkluderad)? I det översta högra hörnet? Har ni någon prioriteringsordning för textplacering kring punkter?**

Det är svarat ovan, för den tidigare frågan. Cycle hire-symboler ska helst placeras i 90 graders vinkel från vägen den tillhör, men annan vinkling är tillåten om så behövs. De får placeras över vägar och gator om situationen kräver det, men det ska undvikas. Cycle hire-symboler får skymma byggnader om det behövs. Busstationer får inte placeras på vägar.

**Vilka typer av punktobjekt representeras bara av sin symbol/text, dvs. själva punkten finns inte själv med på kartan (t.ex. punkter för busshållplatser ska ej vara inkluderade)?**

Det finns i princip inga punktobjekt som representeras av en punkt. Istället placeras dess ikon eller symbol över platsen, alternativt (i undantagsvis) vid sidan av platsen och med en linje som visar punkten plats, vilket är fallet med cycle hire-symboler. Eftersom cycle hire-symbolerna har en inbyggd linje som pekar på platsen, så måste de vara på den platsen. Symbolen tillåts bara rotera runt platsen om den måste flyttas. Symboler utan sådan linje kan flyttas mer fritt. Ikoner, t.ex. bussikoner, får inte flyttas: de måste vara på sin bestämda plats.

**Placerar ni alla gatunamnstexterna på gatorna, så att de följer gatans form (d.v.s. att texten är kurvad längs gatan)? Har ni samma regler för alla typer av gator och vägar, eller har ni olika placeringsregler för olika typer av gator? Placeras alla gatutexter helst centrerat på gatan?**

Gatunamnstexterna placeras alltid på och inom gatan de tillhör. Om en text inte får plats längs sin gata så sätter man texten på två rader inom gatan, om det är möjligt, annars förkortar man texten. Det finns två olika textstorlekar för gator och vägar, beroende på gatan eller vägens storlek och klasstillhörighet. Detta finns definierat i Transport for Londons dokument. Squares, eller torg, kan också betraktas som linjeobjekt, och dess text placeras horisontellt på torget.

**Vilka andra linjeobjekt utöver vägar och gator är det som har text (järnvägar, åar, floder)? Ska dessa linjetexter ha samma placeringsregler som vägar och gator?**

Utöver vägar och gator så är det åar som också har tillhörande text, och de har liknande placeringsregler. QGIS texter har inte geometri kopplat till text, så som ArcMap har. Detta gör att man inte kan kurva texter manuellt i QGIS. Utöver det så blir texterna bokstav för bokstav i efterprocesseringen, när de exporteras från QGIS. ArcMaps texter (s.k. annotation) är fasta geometrier som genereras från deras textsättning (labelling). Hela texten för annotation (inkluderande mellanslag och ev. radbrytning) är sammanbunden. Texten kan justeras genom att dra den utefter linjen om man vill placera om den. En kurvad text är kurvad i QGIS så länge man inte försöker kurva om den eller flytta på den, för då blir den rak. Vid export till PDF, då vi öppnar filen i Illustrator, blir texter bokstav för bokstav. Kolla gärna om man kan få någon slags geometri från en kurvad bokstav

**Angående större floder som *Themsen*, hur sätter ni texter till dem? Ska texten vara kurvad efter flodens form, eller vara rak? Kan den vara horisontell trots att floden inte är det?**

Man skulle kunna skapa linjer i Themsen som används för att sätta text. Just nu skapas texterna i efterprocesseringen manuellt, genom en enkel kopiera-klistra in-metod. Just nu görs Themsen-texten i efterbehandlingen, så det är ingenting som behöver inkluderas i det här projektet. Men du kanske hittar en bättre metod. Themsen är en polygon. Themsen-texten är oftast rak och följer Themsen, och den är alltid i versaler.

**Kan Maplex avgöra om det är bäst att ha en text vänster- eller högerjusterad? Eller genererar den bara en typ av justering, så att ni kan redigera justeringen senare om så behövs?**

Nej. Det får göras manuellt. Testa i QGIS, men det kan vara svårt. Attributtabeln kan definiera hur text ska justeras (från fall till fall). Texter, symboler, och ikoner är olika justerade beroende situation: centrerade, högerjusterade, eller vänsterjusterade. I QGIS har placeringsalternativet *Cartographic* oftast fler valmöjligheter än till exempel *Around point*, men det är ändå begränsade val.

**Vilka typer av kartobjekt (t.ex. gator, byggnader, parker) tillåts bli skymda eller överlappade av andra texter och symboler som inte tillhör de objekten? Vilka objekt tillåts inte alls bli skymda? Kan objektens karaktär på de platsen där de är skymda spela roll för om de får skymmas/överlappas? Finns det olika regler för det, t.ex. om det är tydligt att ett linjeobjekt fortsätter under objektet som skymmer det så är skymning tillåtet, men om det däremot är svårt för kartanvändaren att tolka hur den skymda delen ser ut så är överlappning eller skymning inte tillåtet?**

Inga texter, symboler, eller ikoner får tas bort. De editeras efter placeringen så att de blir synliga. Vid behov kan en skapa en linje som pekar på etikettens egentliga plats. Det godkänns att en icke relaterad text delvis skymmer en byggnad, så länge det syns vilken byggnad varje text tillhör, och att den tillhörande texten till byggnaden finns med. Cykelsymboler och liknande symboler och ikoner får överlappa byggnader om så behövs. Stationssymboler ska alltid placeras på sin punkt eller yta, medan den tillhörande texten (t.ex. en stationstext) kan flyttas runt sin symbol för att hitta en optimal placeringslösning.

**Vilka texter och symboler kan ibland få skymma eller överlappa andra texter och symboler? Jag har noterat att borough- och villages-namn som *Waterloo* och *South Bank* tillåts skymma eller överlappa andra mindre texter, som tillhör byggnader. Finns det andra texter eller symboler som får skymma texter och symboler?**

Eftersom texter inte får tas bort kan ikoner och symboler i vissa fall tillåtas skymma texter, t.ex. gatutexter. Men det är främst de stora, transparenta neighbourhood-borough- och villagetexterna som får överlappa annan kartinformation.

**Hur gör ni valet att en text eller symbol måste tas bort från kartan p.g.a. den blockerar eller skymmer viktig kartinformation? Var går gränsen för att ta bort texter och symboler? Använder ni ett verktyg för tolerans i Maplex?**

Ikoner får inte tas bort, för att de visar var de olika sakerna är lokaliserade. Texten kan förkortas eller sättas på fler rader om nödvändigt, men det ska heller inte tas bort. Tolerans har inte använts i Maplex eftersom all annotation editeras manuellt, men man kan undersöka den saken i PAL.

**Om en hittar ett sätt att kontrollera toleransen för texter (dvs minimumdistansen mellan text och annat objekt – blockera text om distansen är mindre än toleransen), hur ska den toleransen sättas? Har ni en tumregel (t.ex. att större text och större betydelse behöver större toleransavstånd), eller använder ni specifika toleranstal för varje enskild objekttyp?**

Tolerans för avståndsvillkor mellan texter är inte prioriterat. Stora namn är ändå tydliga genom sin design (de har en fet stil, tydlig färg m.m.), och mindre texter, som gatunamn, stör inte nödvändigtvis för att de är nära: de är inte lika iögonfallande.

**Vilka enheter använder ni för att ställa in textsättningsparametrar? *Points* eller *Map Units*? Är det olika i olika fall?**

QGIS gillar *Points*, för att exporten blir bra då, men *Points* är svårare att arbeta med i programmet. I editeringen vill man alltid använda *Map Units*. När man ska producera kartan arbetar man dock med *Points*. Så använd *Map Units*, för vi hanterar QGIS exportproblem själva.

**Har ni olika prioriteringskategorier av texter och symboler, kategorier som talar om hur viktiga de är att inkludera? Om så är fallet, hur appliceras detta vid symbol- och textsättning?**

Nej. Alla texter, symboler, och ikoner ska vara med, inget ska tas bort. En generalisering av kartobjekten och deras geometri har redan gjorts.

### **Använder ni Maplex för att placera både texter och symboler/ikoner, eller bara för textsättning?**

Maplex används bara för att producera texterna, och för att skapa annotation av dem. Annotation är en slags features av texter. Dessa har en redigerbar geometri som definierar exakt var texten ska vara så att den inte kan flyttas runt automatiskt av programmet. Annotation är objektets geometri, som man kan flytta, kurva och så vidare. Vid export från ArcMap så är dessa linjer. Denna metod fungerar inte i QGIS, där kurvade texter blir bokstav för bokstav vid export. Hela textens geometri får definieras i ett andra steg i så fall, till exempel genom att skapa linjer eller ytor.

### **Vad är den bästa ordningsföljden för sättning av texter och symboler? Placeras texterna ut först i Maplex, och efter det symbolerna och ikonerna? Placeras de viktigaste komponenterna ut först? Tar ni lager för lager, eller någon annan ordningsföljd?**

Ikonerna och symbolerna sätts ut först, sedan placeras texterna i efterhand för de ska förhålla sig till symbolerna och ikonerna. Retail units-ikonerna, toalettikonerna, och stationsikonerna representeras först av nummer som tillhör den övriga texten. Numren byts ut mot sina motsvarande ikoner i efterbehandlingen. Detta är bra för då vet man att avståndet mellan texten och ikonen är korrekt. Ikonerna och symbolerna för taxi, parkering, bus stops, cycle hire, och post office är punktbaserade. Ticket stop-symbolen och symboler med pilar kommer från TrueType-font.

### **Vilka är de två skalorna in använder för kartproduktion?**

Den aktuella skalan, för Finder-kartorna, är 1:2250. Översiktskartorna är i skala 1:11 500.

**Hur hanterar ni texter och symboler som efter automatiserad placering hamnar delvis utanför kartan (se figur B1)? Väljer ni att behålla dem så, tar ni bort dem, eller redigerar ni dem manuellt? Kan Maplex identifiera sådana texter och andra felplacerade texter, eller söker ni efter dem manuellt? I så fall, hur identifierar Maplex felaktigt placerade texter och symboler?**



*Figur B1: En text som har placerats delvis utanför kartramen. Från © T-Kartor Group AB.*

Annotation, alltså varje texts placering, är redan definierat för hela London, vilket gör att det inte behöver ändras så mycket i efterbehandlingen. I sådana fall identifieras problem av redigerare, som ändrar placeringen manuellt i Illustrator: man flyttar eller raderar texter, symboler, ikoner eller andra kartobjekt såsom byggnader, som ligger på kanten av kartan. Maplex används bara till att textsätta (labelling). Man textsätter i olika omgångar, klass för klass, p.g.a. att programmet inte klarar av för många på en gång. Annotations skapas från dessa texter (labels). Det finns då många krockar och dåliga placeringar av texter. Alla måste i princip omplaceras. Man kan inte gå tillbaka till Maplex från Annotation.

**På platser med detaljhandel och köpcenter så representeras ikonerna av siffror, till exempel att en etta står för väskikonen. Det är först i efterprocesseringen som numren byts ut mot ikonerna, eller hur?**

Ja, numren byts ut till ikoner i efterprocesseringen, precis som för stationssymbolerna.

**Betyder det att jag bara ska få numren att hamna på en egen rad i textsträngen, och att få texten välplacerad med ikonerna rakt över platsen, och att den**



**resterande delen sker i Illustrator? Hur många nummer/ikoner tillåts på en rad, och hur många tillåts totalt för en textsträng?**

Retail units ligger alltid på egen rad under texten. Det är ofta centrerat, men inte alltid. Ikoner sätts in först i Illustrator för att en ska veta att ikonerna är bra placerade i förhållande till texten. Det finns ingen maxgräns för antalet ikoner för en plats, eftersom antalet redan är fördefinierat för varje plats. Retail-ikonerna ligger alltid samlade på en rad. Men toalettikonen, som är större, behöver inte nödvändigtvis ligga på samma rad som den andra informationen (till exempel en 24 h-symbol).

**I figur B2 nedan verkar det som att ikonerna är redigerade i olika stadier (taxiikonen är färdig, men inte stationssymbolen eller de andra ikonerna). Är taxisymbolen hanterad innan efterprocesseringen, och ändras de andra symbolerna senare i Illustrator (t.ex. att cykeln läggs till i bubblan i Illustrator)? Hur funkar det och hur påverkar det textsättningsprogrammet?**



*Figur B2: Symboler och ikoner som är redigerade i olika stadier. Från © T-Kartor Group AB.*

Vektor-lagren för baskartan har alltid en bestämd ordningsföljd, men den är nödvändigtvis inte samma för texterna och symbolerna. De kan renderas i en annan ordning, som figuren här visar. Men man kan säga att en måste göra textsättningen flera gånger. Efter textsättning får man justera objekt manuellt – sen testa textsättning igen – sen justera manuellt, osv, för att få bra resultat.

**Använder ni FontForge eller annat program för att redigera tecken eller för att skapa nya? Hur går det ihop med att ni bara kan använda de symboler och ikoner som Transport for London har bestämt? Vilka är skälen till att modifiera tecknen? Är inte saker som form och avstånd mellan bokstäver redan definierat i typsnittet, eller tillåts ni redigera sådant för att få bättre resultat?**

I ArcMap använder vi några få TrueType-ikoner, men alla TrueType-ikoner och siffror byts ut mot Illustrator-bilder (ikoner) i efterbehandlingen. QGIS kan hantera SVG-bilder bra. Vi har inte använt SVG-bilder, men en kan testa det i QGIS. De ser bra ut visuellt, men dess geometri är troligen kvadratisk, medan TrueType kan editeras mer noggrant gällande objektens geometri. Fördelen med att ha ikoner som TrueType-font är således att de agerar mer som vanliga texter när det gäller placering och förmåga att blockera andra objekt. Det betyder att övriga objekt borde se dessa ikoner mer som texter, vilket är bra. Nackdelen och utmaningen är att ha ikoner med fler lager och färger.

**Kommer transparens, färg och storlek av text behövas programmeras i PAL också? Eller är all den informationen redan definierad i fonterna som används, som Transport for London valt ut? Eller sker den designen efteråt i Illustrator?**

Transparens görs i efterprocesseringen: ArcMap klarar inte av det på ett bra vis. Så transparens behövs inte definieras i QGIS. Det kan göras i QGIS, men det blir oftast inte bra vid exportering, och då behöver det ändå ändras i efterprocesseringen.

**Kurvade texter behandlas bokstav för bokstav i QGIS. Är det något som jag ska ha i åtanke för textsättningsprogrammet? (Ska jag bara låta de kurvade texterna vara som de är när de genererats?)**

Om kurvade texter flyttas manuellt i QGIS så blir de automatiskt raka. Då måste man ändra rotationen av texten, men de går inte att göra kurvade igen. Texter i QGIS har inte en geometri på samma sätt som i ArcMap. En kan testa att i QGIS skapa nya linjer eller punkter som definierar hur gatunamnen ska vara placerade, för att undvika manuell omplacering av texter.

Texter är tydligare när de är raka, därför är raka texter att föredra. Men om gatan är väldigt kurvad, så kan texten inte vara rak eftersom de måste följa gatans form. Dock skulle många gator som endast är lite kurvade ha raka texter. Man kan även ta bort noder i till exempel linjer för att generalisera objekt. Vägar och gator är uppdelade i Major, Minor, osv, uppdelade i olika kategorier. Sen finns det trappobjekt. Dessa är representerade av en grupp linjer: en linje är ett trappsteg.

**Angående stationer, sätter ni alltid stationssymbolerna över stationstexten (se figur B3)?**

#### Stations



Figur B3: Placering av stationstexter och stationssymboler. Från © T-Kartor Group AB.

Placeringen av stationstexten kring stationssymbolen varierar beroende på situationen. Men symbolen ska alltid vara på sitt objekt. Texten ska ligga jäms med symbolerna både i x- och y-led. Detta är ett problem i ArcMap och QGIS. Det är därför de byts ut i efterbehandlingen, där vi har mer kontroll. Detta är ett stort problem för oss. Det optimala vore att ha en punkt där stationen är, och från den punkten rendera ut ikoner och text så att allt har rätt storlek och avstånd. Detta kanske kan göras med funktionaliteten *Rule-based Labeling* i QGIS.

**För handelsplatser (eng. retail units), sätter ni alltid ikonerna under texten (se figur B4)?**



Figur B4: Ikonplacering för handelsplatser. Från © T-Kartor Group AB.

Retail unit-ikoner placeras alltid under texten. Ikonerna har valts ut av kund och är aldrig fler än 4 stycken. Landmark building kan ha retail unit-ikoner, då är de vänsterställda. Neighbourhood kan också ha retail unit-ikoner, men de är centrerade.

### **Hur genereras övergångsställena? Transport for London har definierat hur de ska se ut, men hur görs det i praktiken?**

Övergångsställen genereras bäst genom att skapa en yta. Detta är bättre än att använda en linje, för då vet man utsträckningen av övergångsstället, så att textöverlappning kan undvikas. Men det är inte prioriterat.

### **Hur skapar ni YAH-cirkeln till kartorna? Baseras det på punktlagret *LMF YouAreHere* (YAH)? Hur görs det i Maplex? Cirkeln kanske kan genereras med hjälp av QGIS' verktyg *Geomerty generator*?**

Den aktuella cykelstationsikonen tas bort och ersätts med YAH-cirkeln och pilen i produktionen. YAH-punkten är alltid placerad en tredjedel från botten av kartan och två tredjedelar från toppen av kartan. Byggnader och texter kan ibland tas bort om de är långt ifrån YAH-punkten eller om de är i kanten av kartan, därför att de inte är lika viktiga som annan information. Den kartinformation som är framför kartanvändaren (d.v.s. framför YAH-punkten) är viktigast. I QGIS kan YAH-cirkeln i nuläget inte blockera andra objekt. En får i så fall skapa en linje med samma tjocklek som YAH-cirkeln för att blockera, alternativt en yta eftersom linjer inte kan blockera objekt i QGIS. Men detta är mindre prioriterat i sammanhanget.

**Hur påverkar YAH-cirkeln textsättningen? Finns det några objekt som inte får finnas inom YAH-cirkeln? Finns det objekt som i vissa fall får överlappa YAH-cirkeln, till exempel borough-namn eller andra större texter eller symboler?**

Placeringen av cirkeln gör att många texter, symboler, och ikoner behöver flyttas när man skapar produkter, viss flyttning och justering sker nu i efterbehandlingen. Inga texter, symboler, eller ikoner får överlappa YAH-cirkeln.

**Hur fungerar rotation av karta? Roterar man kartan när den redan innehåller texter och symboler, eller genererar man texterna och symbolerna alltid efter rotationen har gjorts? Kan man ha kvar vissa texter efter rotation, t.ex. gatutexter, medans punkttexter såsom texter till byggnader kräver redigering efter rotation?**

Vi har en databas med texter och ikoner där de är placerade så att norr är upp på kartorna. För att skapa produkter roteras kartan så som behövs. Symboler och raka texter placeras horisontellt. Upptochnervända gatutexter flippas så att de blir läsbara. Dock så är metod för textbehandling vid rotation av karta ingen prioritet för det här projektet.

**Använder ni någon sorts dynamisk styckning (eng. truncate) i Maplex, som vid behov kan förkorta texten vid kartrotation eller vid andra situationer, eller görs förkortningen manuellt? Har ni en gräns för hur mycket orden tillåts att förkortas?**

Text får inte förkortas hur som helst. Till exempel får inte *the* stå själv på en egen rad. Om en text inte får plats ska den förkortas, alternativt delas upp på fler rader.

**Hur görs uppdelning av text till två rader? Används alltid #-tecknet som uppdelare i attributtabeln?**

#-tecknet kan användas för att dela på text. Det finns ett annat sätt där man använder en fil som definierar hur text ska delas upp, men det är inte testat. Det kan undersökas i QGIS, hur det kan göras.