Label Placement for Outliers in Scatterplots

H. Mumtaz †[1], M. van Garderen[2], F. Beck[3], and D. Weiskopf[1]

[1]VISUS, University of Stuttgart, Germany
[2]University of Konstanz, Germany
[3]paluno, University of Duisburg-Essen, Germany

Abstract

In many application scenarios, outliers can be associated with specific importance for various reasons. In such cases, labeling outliers is important to connect them to the actual semantics of the respective entity. In this paper, we present a cost-based greedy approach that places labels with outliers within scatterplots. The approach uses a search strategy to find the position that represents the least cost to place labels. Our approach can also produce different labeling outcomes by adjusting the weights of the criteria of the cost function. We demonstrate our approach with scatterplots produced from object-oriented software metrics, where outliers often relate to bad smells in the software.

CCS Concepts

• ***Human-centered computing*** → ***Visualization; Visualization application domains; Information visualization;***

1. Introduction

Outliers are individual data points that discern themselves from the rest of the data with respect to a relevant measure. They can be important in data analysis for various reasons, for instance, because they indicate remarkable individuals (both good and bad) or problems with data recording and processing. Whereas visualization research has discussed clusters (i.e., groups of similar data points) in detail already, we observe that the visual representation and marking of outliers have not yet received the same level of attention.

This work studies the particular problem of labeling outliers in scatterplots. It is motivated by the analysis of code quality metrics in software projects. We use scatterplots to find problematic data points that show up as visual outliers. Labels are crucial in this scenario because they connect the data points to the semantics of the respective data entity. However, we did not find an existing approach that properly handles this scenario, which is different from labeling applications in previous work: (i) data points that need to be labeled are in low-density areas and (ii) non-outlier points exist as potential obstacles that should not overlap with labels.

We suggest an algorithm for this outlier labeling problem that places the labels next to the point or, if the label needs to be placed at some distance, it draws a leader between the label and the point. Various criteria are considered such as the distance of the label to the point or different kinds of overlap between labels, leaders, and points. Each criterion is weighted (to adjust its influence on the labeling layout) and linearly aggregated into the overall cost function. We demonstrate the practical applicability of the approach with object-oriented software metrics data, where outliers are related to bad smells. Bad smells are introduced because of inappropriate design or implementation decisions during software development [Fow18]. Moreover, we show how the weights of the criteria of the cost function can be tuned to optimize for different labeling results. Figure 1 shows a labeling result for the outliers (depicted as ●) regarding the two software metrics *coupling between objects (cbo)* and *response for class (rfc)*.

2. Related Work

The automated placement of labels on maps and other visualizations has been studied for decades. Most variants of the labeling problem are NP-hard [FW91, MELS95], but many heuristic approaches have been proposed [WS09, DKMT07]. Most of these approaches place labels either adjacent to the points or around the perimeter of the visualization, but for our application, methods that combine adjacent and distant labels, are of more interest. The particle-based labeling method by Luboschik et al. [LSC08] is a fast heuristic that uses adjacent labels for as many points as possible and distant labels for the remaining points. A drawback of this approach is that it may result in many crossings between leaders. The clutter-aware labeling approach by Meng et al. [MZLL15] places labels with minimal visual clutter, as specified by a cost function. However, when an area is considered too cluttered, labels are simply left out. To the best of our knowledge, none of the existing methods aim specifically at labeling the outliers of a visualization.

Several techniques—such as minimum spanning tree [JTS01, LYCG08], *k*-means clustering [PDN11,IH93], and *k*-nearest neigh-

† Corresponding author

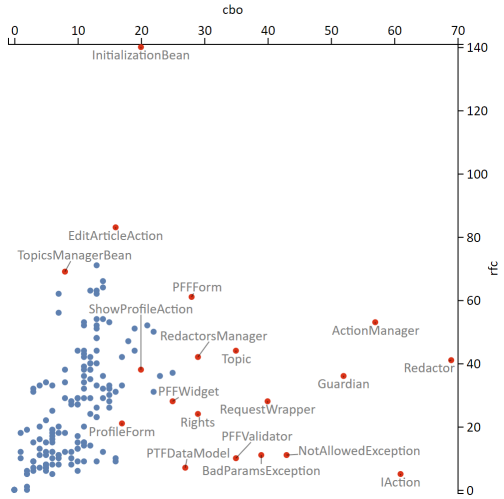

Figure 1: *Labeling outliers ● in a scatterplot of* coupling between objects (cbo) *and* response for class (rfc). *Non-outlier ● classes are not labeled.*

bors [HKF04, ABP06, AP02, Agg15]—are employed to detect outliers. We adopt the k-nearest neighbors approach to detect outliers in scatterplots. The k-nearest neighbors technique discern outliers by ranking the data points in terms of distance from the k nearest neighbors. Some techniques use visualizations for outlier analysis, for instance, Bernard et al. [BDSF17] present a visual interactive system for analyzing outliers. Their system also allows users to visually compare the results of different outlier analysis algorithms.

3. Algorithm

We propose a cost-based greedy approach that searches for the best possible position to place labels with outliers. The approach places one label after the other without changing the layout of already placed labels.

Outliers are determined using the k-nearest neighbor method as an estimation of local density. We use the Euclidean distance to compute the distance between a point and its k nearest neighbors. If this distance is larger than a threshold, the point is classified as an outlier. We apply the Euclidean distance in image space so that the impact of non-normalized data on distance computation can be minimized.

Let N be the set of n points in the visualization, and let $M \subseteq N$ be the subset of m points (outliers) that should be labeled. The input consists of N and M, with coordinates (x_i, y_i) for each point $p_i \in N$. Furthermore, each point $p_i \in M$ has an associated label l_i with dimensions $w_i \times h_i$. The output should consist of coordinates (x'_i, y'_i) for each label that could be placed.

3.1. Cost Function

A cost function is used to compare placement options and the alternative with the lowest cost is chosen. We define the cost $C_i(x', y')$ as a linear superposition of four terms—$C_{overlap}$, C_{buffer}, $C_{distance}$, and $C_{position}$ (see Table 1)—for placing label l_i at position (x', y'). Each term is the weighted linear aggregation of some criteria, for instance, to achieve legibility of all labels, three types of overlap are defined in $C_{overlap}$: First, the number of points in M (outliers) that are covered by l_i; second, the number of points in $N \backslash M$ (non-outliers) covered by l_i; the third takes into account the number of previously placed labels covered by l_i to prevent label-label overlaps. The influence of each term of the cost function on the labeling layout can be controlled by adjusting the weights of the underlying criteria defining the term.

Table 1: *Terms of the cost function* $C_i(x', y')$.

Term	Description
$C_{overlap}$	overlap of the label l_i with outliers M, non-outliers $N \backslash M$, and other labels.
C_{buffer}	overlap of the label buffer of l_i with outliers M and overlap of the leader buffer with outliers M, non-outliers $N \backslash M$, and labels.
$C_{distance}$	distance between point p_i and its label l_i.
$C_{position}$	position of the label within the chart space and with its relative position with respect to the position of the point in the scatterplot.

To avoid confusion about which label belongs to which point, leaders should not cross or be very close to other points or labels. To achieve this, we define a *leader buffer* as the area directly surrounding the leader, as shown in Figure 2. Furthermore, a label that is placed in close proximity to other points in M might make it unclear to which of them the label belongs. Therefore, we also introduce a *label buffer* (30% of the label height) around the bounding box of label l_i. C_{buffer} uses label and leader buffers to include four criteria: the first relates to the overlap of the buffer of label l_i with the number of points in M; the second incorporates the overlap of the leader buffer with the number of points in M; the third checks the number of points in $N \backslash M$ contained in the leader buffer of l_i; and the fourth is concerned with the number of labels intersecting the leader buffer of l_i.

The leader is constructed by connecting p_i to the center of l_i and cutting the line segment off at its intersection with the bounding box of the label, as illustrated in Figure 2. $C_{distance}$ represents the length of the leader.

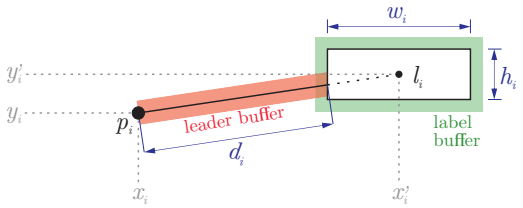

Figure 2: *Point* p_i *at position* (x_i, y_i) *with its label* l_i *of size* $w_i \times h_i$ *at position* (x'_i, y'_i)*, with distance* d_i *between* p_i *and* l_i.

Finally, it is desirable to place labels only within a pre-defined chart area and in a position where they are less likely to later be in the way of other labels. To include these criteria, we define $C_{position}$ in our cost function. Since we are labeling outliers, we consider a model where we prefer to place the labels toward the outside of the visualization. This means that the preferred relative position of the label with respect to the point depends on the relative position of the point in the overall visualization. If the point is in the left half of the visualization, we prefer to place the label to the left of the point, and for points in the right half of the visualization, we prefer to place the label to the right of the point. Similarly, we consider the top and bottom relative positions. It is also undesirable to place labels far outside the chart area, so the respective criterion of $C_{position}$ restricts the labels to be in a pre-defined area of the scatterplot.

3.2. Search Strategy

To find the best position of a label for each outlier, we need to evaluate the cost function at the potential label positions. Since an exhaustive search among all possible pixel positions would be computationally expensive, we only check a sample of promising locations. For this, we consider a circular range around the position to label (points outside this area are unlikely to produce low costs when taking into account the leader length). More specifically, we use a grid-based search strategy based on a radial grid with its center at the outlier position. Incrementing the radius by 30% in each iteration, we vary the angle in 100 steps per radius. We stop after 20 iterations and label the point with the label that produced the lowest cost among the searched positions. While this option produced good results in acceptable time in our experiments, other search heuristics are applicable and might be more efficient. Finding the best search heuristic, however, was out of scope for this paper.

4. Application Example

We use object-oriented software metrics to draw scatterplots and label classes of the *redactor*[1] software project. Each data point in the dataset represents a software class and the label is the name of the class. Mumtaz et al. [MBW18] investigated the connection between outliers and bad smells. They observed that software classes that are depicted as outliers in visualizations are potentially associated with a higher probability of carrying bad smells. We take this connection of outliers and bad smells as a motivation to label software classes that are depicted as outliers in scatterplots. Figure 1 and Figure 3 show outliers with labels in two different scatterplots. The scatterplots are built with *coupling between objects (cbo)* in two different combinations: one with *response for class (rfc)* and the other with *weighted method per class (wmc)*. These software metrics are commonly used to detect bad smells in software projects [OKSB13]. For instance, high *weighted method per class (wmc)* value indicates the possibility of the existence of the *large class* bad smell. In Figure 3, classes with relatively high *wmc*—depicted in less dense regions (outlier regions)—have the high probability of carrying the *large class* bad smell, therefore, these classes are important and we label them.

It may be required in an application scenario to change the number of data points that need labels. Our approach provides the control of changing the number of data points by adjusting the threshold value for outliers. In Figure 4, the number of labeled points is increased (in relation to Figure 3) by reducing the threshold value. In Figure 4, we label the maximum number of outliers before we start to see the overlaps. At the moment, it has only one overlap (i.e., a leader is overlapping non-outlier points).

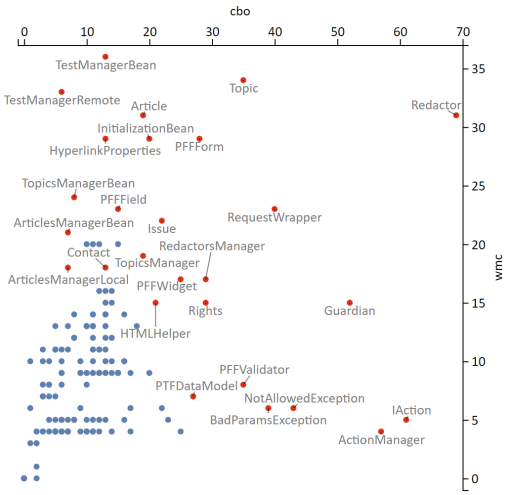

Figure 3: *Labeling outliers for dimensions* coupling between objects (cbo) *and* weighted method per class (wmc).

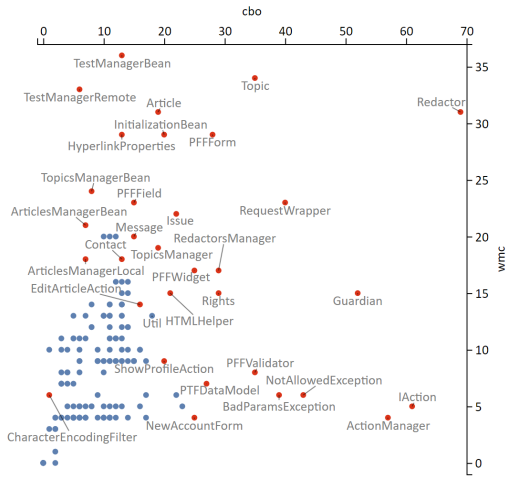

Figure 4: *The number of labeled outliers are increased with respect to Figure 3.*

[1] https://doi.org/10.5281/zenodo.322445

5. Impact of Weights of Cost Function

We tested our approach with multiple software datasets to find the weights of the criteria of the cost function that can be used as default. Although it is difficult to optimize the weights that can work in every scenario, we tried to obtain default values that can generate relatively acceptable labeling results. We tried to prioritize the terms of our cost function so that the weights of relatively important criteria are set to high, for instance, avoiding overlap of labels is more important than reducing leader length. We saw a few scenarios where the default weights could not produce desired outputs—overlaps are observed. In such cases, the weights can be tailored.

We observe a few cases where overlaps hinder in inferring the labeling output. For instance, in Figure 5, a leader is overlapping a label *RequestWrapper*. To remove this, we increase the weight of the label-leader overlap of C_{buffer}. As a result, the overlap of label *RequestWrapper* and leader is removed (shown in Figure 6). However, as a result of this change, a new overlap of non-outliers and label *ArticleDetailsModel* is introduced. We increase the weight of the respective criterion (non-outliers covered by label) in $C_{overlap}$ to remove these non-outliers and label *ArticleDetailsModel* overlap. The resulting output, depicted in Figure 7, shows that the label *ArticleDetailsModel* is moved to remove its overlap with non-outliers.

To show that our method can generalize to arbitrary scatterplots, we produce a labeling result from the data[2] about mental disorders in the United States. It can be seen in Figure 8 that the median income is negatively correlated with the mental disorder, however, there are a few outliers that could explain some unusual behavior. For instance, New Hampshire falls into high-income category, but people from this state still experience high mental disorders.

6. Conclusion and Future Work

In this paper, we presented a cost-based greedy approach that places labels with outliers because we argue that outliers can be of interest for various reasons. The weights of criteria in the cost function can be adjusted to produce different labeling results in different application scenarios. We demonstrated our approach with scatterplots produced from object-oriented software metrics. First, we showed the results specific to a software engineering application, where we labeled outlier classes that are potentially prone to bad smells. We also illustrated the removal of overlaps by varying the weights in our cost function. In our future work, we aim to assess the generalizability of labeling outliers in other point-based visualizations. We also plan to prioritize the order in which labels can be placed.

Acknowledgements

F. Beck is indebted to the Baden-Württemberg Stiftung for the financial support of this research project within the Postdoctoral Fellowship for Leading Early Career Researchers. M. van Garderen was funded by the European Research Council (ERC) under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement n° 319209 (project NEXUS 1492).

[2] https://www.urban.org/research/publication/geographic-patterns-disability-insurance-receipt

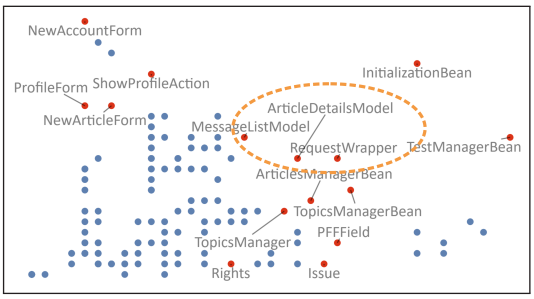

Figure 5: *Label-leader overlap: the leader of the label* ArticleDetailsModel *passes through* RequestWrapper.

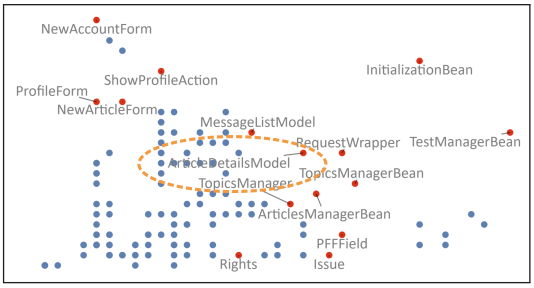

Figure 6: *Label-leader overlap is removed by increasing the weight of the respective criterion in* C_{buffer}*, but the label* ArticleDetailsModel *is now covering non-outliers.*

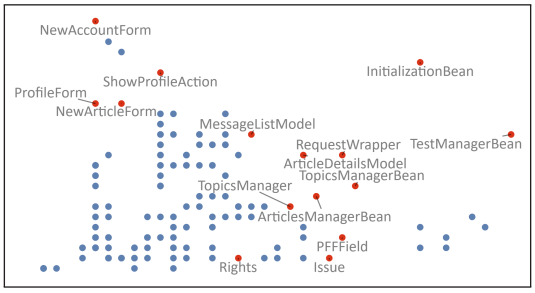

Figure 7: *Label and non-outliers overlap is removed.*

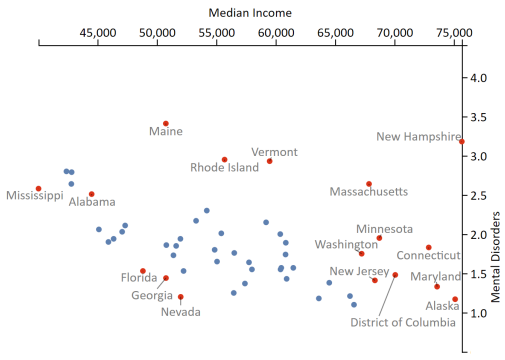

Figure 8: *Negative correlation between* mental disorders *and* median income *in the United States mental disorders data, with a few outliers (e.g., New Hampshire).*

References

[ABP06] ANGIULLI F., BASTA S., PIZZUTI C.: Distance-based Detection and Prediction of Outliers. *IEEE Transactions on Knowledge and Data Engineering 18*, 2 (2006), 145–160. doi:10.1109/TKDE.2006.29. 2

[Agg15] AGGARWAL C. C.: Outlier Analysis. In *Data Mining* (2015), Springer, pp. 237–263. doi:10.1007/978-3-319-14142-8_8. 2

[AP02] ANGIULLI F., PIZZUTI C.: Fast Outlier Detection in High Dimensional Spaces. In *European Conference on Principles of Data Mining and Knowledge Discovery* (2002), Springer, pp. 15–27. doi:10.1007/3-540-45681-3_2. 2

[BDSF17] BERNARD J., DOBERMANN E., SEDLMAIR M., FELLNER D. W.: Combining Cluster and Outlier Analysis with Visual Analytics. In *EuroVis Workshop on Visual Analytics* (2017), Sedlmair M., Tominski C., (Eds.), The Eurographics Association. doi:10.2312/eurova.20171114. 2

[DKMT07] DOGRUSOZ U., KAKOULIS K. G., MADDEN B., TOLLIS I. G.: On Labeling in Graph Visualization. *Information Sciences 177*, 12 (2007), 2459–2472. doi:10.1016/j.ins.2007.01.019. 1

[Fow18] FOWLER M.: *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Professional, 2018. 1

[FW91] FORMANN M., WAGNER F.: A Packing Problem with Applications to Lettering of Maps. In *Proceedings of the 7th Annual Symposium on Computational Geometry* (1991), ACM, pp. 281–288. doi:10.1145/109648.109680. 1

[HKF04] HAUTAMAKI V., KARKKAINEN I., FRANTI P.: Outlier Detection using K-nearest Neighbour Graph. In *Proceedings of the 17th International Conference on Pattern Recognition* (2004), vol. 3, IEEE, pp. 430–433. doi:10.1109/ICPR.2004.1334558. 2

[IH93] IGLEWICZ B., HOAGLIN D. C.: *How to Detect and Handle Outliers*, vol. 16. Technometrics, 1993. doi:10.1080/00401706.1994.10485810. 1

[JTS01] JIANG M.-F., TSENG S.-S., SU C.-M.: Two-phase Clustering Process for Outliers Detection. *Pattern Recognition Letters 22*, 6-7 (2001), 691–700. doi:10.1016/S0167-8655(00)00131-8. 1

[LSC08] LUBOSCHIK M., SCHUMANN H., CORDS H.: Particle-based Labeling: Fast Point-feature Labeling without Obscuring other Visual Features. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1237–1244. doi:10.1109/TVCG.2008.152. 1

[LYCG08] LIN J., YE D., CHEN C., GAO M.: Minimum Spanning Tree Based Spatial Outlier Mining and its Applications. In *International Conference on Rough Sets and Knowledge Technology* (2008), Springer, pp. 508–515. doi:10.1007/978-3-540-79721-0_69. 1

[MBW18] MUMTAZ H., BECK F., WEISKOPF D.: Detecting Bad Smells in Software Systems with Linked Multivariate Visualizations. In *IEEE Working Conference on Software Visualization* (2018), IEEE, pp. 12–20. doi:10.1109/VISSOFT.2018.00010. 3

[MELS95] MISUE K., EADES P., LAI W., SUGIYAMA K.: Layout Adjustment and the Mental Map. *Journal of Visual Languages & Computing 6*, 2 (1995), 183–210. doi:10.1006/jvlc.1995.1010. 1

[MZLL15] MENG Y., ZHANG H., LIU M., LIU S.: Clutter-aware Label Layout. In *IEEE Pacific Visualization Symposium* (2015), IEEE, pp. 207–214. doi:10.1109/PACIFICVIS.2015.7156379. 1

[OKSB13] OUNI A., KESSENTINI M., SAHRAOUI H., BOUKADOUM M.: Maintainability Defects Detection and Correction: A Multi-objective Approach. *Automated Software Engineering 20*, 1 (2013), 47–79. doi:10.1007/s10515-011-0098-8. 3

[PDN11] PAMULA R., DEKA J. K., NANDI S.: An Outlier Detection Method Based on Clustering. In *2nd International Conference on Emerging Applications of Information Technology* (2011), IEEE, pp. 253–256. doi:10.1109/EAIT.2011.25. 1

[WS09] WOLFF A., STRIJK T.: The Map-labeling Bibliography. *http://ill.www.ira.uka.de/map-labeling/bibliography/* (2009). 1