EVALUATING SPATIAL QUERIES OVER DECLUSTERED SPATIAL DATA

A Thesis

Submitted to the Faculty

of

Purdue University

by

Eslam Almorshdy

In Partial Fulfillment of the

Requirements for the Degree

of

Masters of Science

August 2019

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF THESIS APPROVAL

Prof. Vernon J. Rego, Chair

Department of Computer Science

Prof. Aditya P. Mathur

Department of Computer Science

Prof. Christopher W. Clifton

Department of Computer Science

Approved by:

Prof. Voicu Popescu

Head of the Graduate Program

To God, To my family

May this be of use

ACKNOWLEDGMENTS

This would not have been possible without Prof. Rego's academic and personal support. I am grateful to my committee members, Prof. Mathur and Prof Clifton, for their prompt feedback and support. I am grateful to have been supported by Purdue's RCAC, Purdue's OVPR, and Purdue Libraries' Research Data Unit.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Figure Page

ABSTRACT

Almorshdy, Eslam M.Sc., Purdue University, August 2019. Evaluating Spatial Queries over Declustered Spatial Data. Major Professor: Vernon Rego.

Due to the large volumes of spatial data, data is stored on clusters of machines that inter-communicate to achieve a task. In such distributed environment; communicating intermediate results among computing nodes dominates execution time. Communication overhead is even more dominant if processing is in memory. Moreover, the way spatial data is partitioned affects overall processing cost. Various partitioning strategies influence the size of the intermediate results. Spatial data poses the following additional challenges: 1)*Storage load balancing* because of the skewed distribution of spatial data over the underlying space, 2)*Query load imbalance* due to skewed query workload and query hotspots over both time and space, and 3)*Lack of effective utilization* of the computing resources. We introduce a new kNN query evaluation technique, termed BCDB, for evaluating nearest-neighbor queries (NN-queries, for short). In contrast to clustered partitioning of spatial data, BCDB explores the use of declustered partitioning of data to address data and query skew. BCDB uses summaries of the underling data and a coarse-grained index to localize processing of the NN-query on each local node as much as possible. The coarse-grained index is locally traversed using a new uncertain version of classical distance browsing resulting in minimal $O(\sqrt{k})$ elements to be communicated across all processing nodes.

1 INTRODUCTION

In this thesis we answer spatial queries over large volumes of big spatial data. Specifically we evaluate nearest neighbour queries (kNN) [1] queries over large volumes of spatial data. kNN query, for a given focal point, reports k closes objects. The data we consider are spatial (multidimensional). we consider real-life geographic data as well as several data derived from uniform, gaussian, and joint (correlated) distributions. Systems adding spatial awareness to big data processing [2–4] have considered spatial and random partitioning strategies. In Spatial decomposition logical space partitions are mapped to computing nodes. It has two drawbacks. First, under utilization of resources since only partitions relevant to the query can contribute. Second, vulnerability to hot-spots if subjected to skewed query workload. In Random partitioning, every computing node is assigned a random set of objects. While seemingly fully occupied with processing, it is poorly utilized as many local solutions can be ruled as non-contributing after necessary communication (reduction). To address these drawbacks in the context of NN-Queries we propose BCDB (Balanced Coarse Distance browsing); an in situ approach for processing NN-Queries over big spatial data. We utilize declustering to achieve *Load balancing.* To minimize uncertainty of local solutions and consequently communication overhead, a coarse grained index (over leaf MBRs) is shared among computing nodes. The coarse grained index is locally traversed using a new uncertain version of classical distance browsing [5] resulting in minimal $O(\sqrt{k})$ elements to be communicated across all processing nodes.

Section 1.1 discusses properties for distributed evaluation of a database operator. Section 1.2 formulates an abstraction for evaluating kNN and motivates for coarse evaluation of kNN. Chapter 2 reviews R-trees, Nearest Neighbour Queries, Space Filling Curves, and Big Spatial Data Processing. Chapter 3 introduces preliminaries and the details of the proposed coarse distance browsing. Chapter 4 analytically and empirically estimates bounds on uncertainty in the output of coarse distance browsing. Appendix A shows a visualized execution trace of an instance of coarse distance browsing.

1.1 Operator Taxonomy

When considering distributed evaluation, distributiveness is the first examined property of an operator. For operators that can be recursively expressed we differentiate whether it is a generative or a structural expression.

1.1.1 Distributiveness

Distributive operators (those that can be realized as a select/filter), while inherently parallel and show no need for communication, are sensitive to skewed query workload if no careful partitioning strategy is adopted (*load balancing challenge*). For example. spatially partitioning spatial data and subjecting it to a spatial query workload will result in hot spots.

1.1.2 Structural vs Generative recursion

An expression is structurally recursive when the input's structure is not altered through the recursive step. For example, traversing a tree or searching a sorted array doesn't alter the tree's hierarchical composition or the array's sortedness. Generative recursion happens when the structure of the input is altered ahead of the recursive step. For example, ordering around the pivot of a recursively defined quick sort. Operators that can be expressed using structural recursion [6] are distributive by definition. Operators that can only be expressed using generative recursion (i.e., input to recursive calls is different from original input's structure or order) are non distributive and incur communication overhead (*communication & load balancing challenge*).

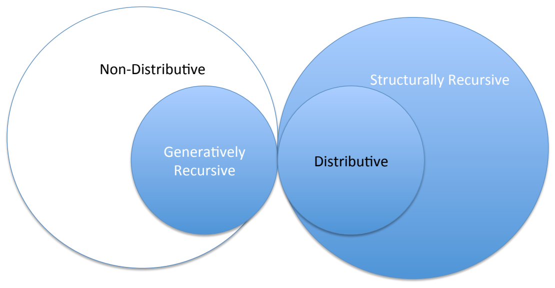


Figure 1.1.: Distributiveness and kind of recursion

1.2 Intuition

We chose kNN as the most common non-distributive spatial operator, so it is representative of generatively recursive operators (*communication challenge*). Similar to distributive operators kNN have a load balancing challenge. Further, kNN is a building block for other NN operations. For a point space $X = \cup_{i=1}^{n}(x_i)$ where $|x_i| = \frac{|X|}{n}$; a distributed implementation of kNN_{select} can be expressed as

$$kNN(\cup(x_i)) = \underbrace{kNN}_{\text{reduce}}(\cup(\underbrace{kNN(x_i)}_{\text{map}})) \tag{1.1}$$

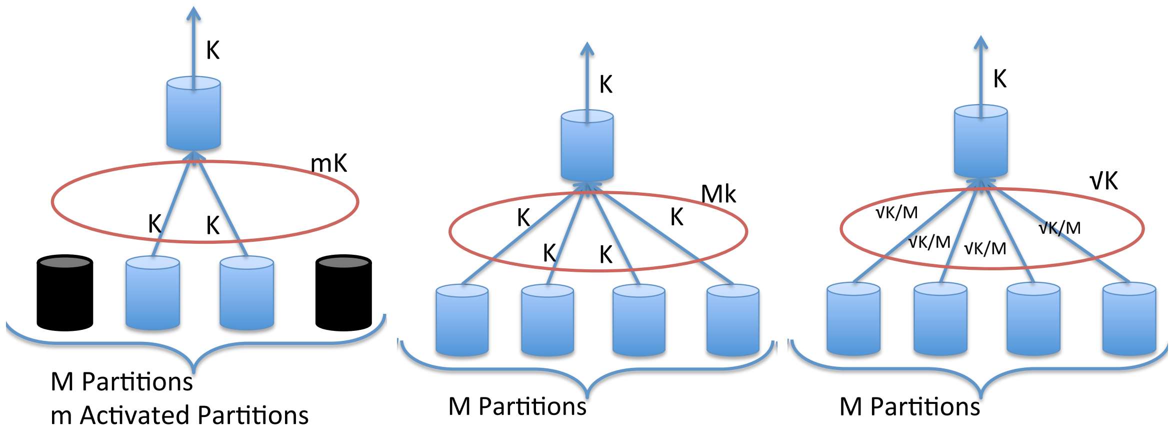
In the above expression, reapplication of kNN over intermediate results (k from each partition) is necessary for correctness as we have no assertion on the quality of any of the intermediate results.

1.2.1 Communication cost

Despite advances in secondary storage and growing capacities of main memory, communication dominates total execution time [7]. Communication is orders of magnitude slower than main memory. Such slowness warrants investigating enhancing intermediate results and deferring communication as much as possible. Below we discuss cost for different partitioning strategies for kNN operation as expressed in the generatively recursive equation (1.1).

Communication cost is sensitive to data partitioning strategy. For a kNN operation under random partitioning intermediate results from all processing nodes are to be further processed. kNN under spatial Partitioning only intermediate results of nodes near focal point is processed while other nodes are idle w.r.t to that single query.

> Example: For in memory kNN over a point set of size X, $k = 0.0001X$, Number of partitions M=1000, node capacity $= X/M$. Communication cost of random partitioning $MK = 0.1X >$ node capacity. This can degrade to actual sorting of the whole data set.



(1) Spatial (2) Random (3) This Work

Figure 1.2.: Size of intermediate results and number of active nodes for spatial and random partitioning strategies

1.2.2 Modeling Uncertainty for a single operator

In a generatively recursive decomposition, intermediate solutions, as-is, posses uncertainty. Communication to a reduce phase resolves such uncertainty. For example, in the case of kNN, an abstraction over uncertainty of intermediate solutions can be expressed as 3 totally ordered classes of partially ordered elements:

$$\left.\begin{array}{l}
\bullet\ \text{Contributing set of size 0 to } k\ \textit{(partially ordered)} \\[1em]
\bullet\ \text{Uncertain set of size 0 to } k\ \textit{(partially ordered)} \\[1em]
\bullet\ \text{Pruned set of size } \leq |x_i|\,[-k]\ \textit{(partially ordered)}
\end{array}\right\}\text{Totally ordered}$$

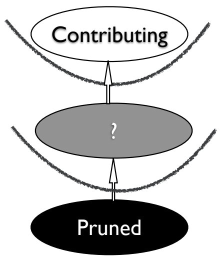


Figure 1.3.: Totally ordered classes of intermediate results in absence of communication

1.2.3 Enhancing intermediate results

Below we discuss an approach to enhancing the quality of the intermediate results. Consider figure 1.4 points divided in 2 partitions red and blue. When evaluating kNN locally, points can be sorted (w.r.t. proximity to a focal point). If we further share a statistic across partitions (e.g., for a point we have a count of proximate points that exist in whole point space, not just the partition), this will allow giving weight to local solutions and will decrease uncertainty. See figure 1.4

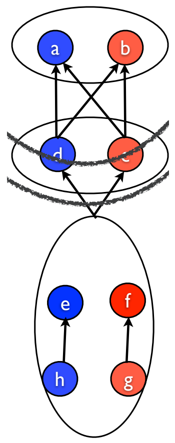


Figure 1.4.: Enhancing intermediate results across 2 partitions. Enhanced with a statistic (count), red and blue partitions produce better intermediate results for kNN evaluation: For $k = 2$ red partition reports $\{b\}$ as certain, blue partition reports $\{a\}$ as certain. For $k = 3$ red partition reports $\{b\}$ as certain and $\{c\}$ as uncertain; blue partition can report $\{a\}$ as certain and $\{d\}$ as uncertain. For $k = 4$ red partition reports $\{b, c\}$ as certain and none as uncertain, blue partition reports $\{a, d\}$ as certain and none as uncertain.

1.2.4 Output Sortedness and Early reporting

In order based operation similar to kNN; order among elements of the result is not part of the contract. E.g., for kNN, the resulting k elements needn't be sorted. This characterization is beneficial as it allows early reporting of solutions. In other words, the certain set in figure 1.3 can remain in partial order.

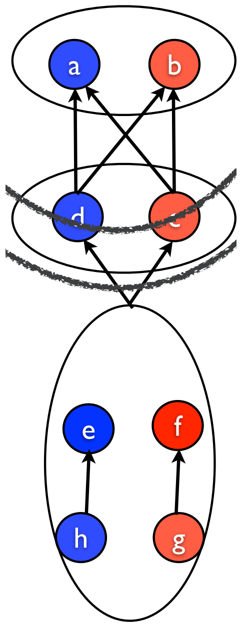


Figure 1.5.: Early reporting: Blue and Red are 2 partitions. Evaluating kNN For k=3 blue partition reports $\{a\}$ as certain , and red partition reports $\{b\}$ as certain, without the need to determine/resolve order among a and b

2 LITERATURE REVIEW

2.1 R-trees

R-tree [8] is a hierarchical data structure based on B+ tree [9]. R-Tree is used as a spatial access method (indices to store/retrieve geometric objects). An object of d dimensions is represented by a d-dimensional Minimum Bounding Rectangle (MBR). Every internal node is an MBR for its underlying MBR(s). leaf and internal MBRs can overlap. A lot of variations have been developed. R+trees [10] avoids overlapping MBR(s) thus allowing for faster point containment queries. R*trees [11] optimizes for minimum overlap, area covered, margins, and storage. Hilbert-R-trees [12] are B+ trees over the hilbert value of the object's centroid. Hilbert space filling curve (as we discuss later) is a proximity preserving linearization for d-dimensional space. Bulk loading variations were introduced: Packed R-Tree [13], Hilbert Packed R-tree [14], Sort-Tile-Recursive R-Tree [15]. For a comprehensive survey or R-Trees and it's applications, see [16].

2.2 Nearest Neighbour Queries

k nearest neighbours (kNN) were first introduced by [1]. Given a focal point kNN finds the k nearest objects to the focal point. [1] developed pessimistic and optimistic distance metrics for object containment in an MBR. [1] devised a branch and bound algorithm to minimize number of visited nodes in an R-tree. [5] Introduced distance browsing: an incremental algorithm to evaluate kNN. Distance browsing is essential when adding a relational predicate to the kNN query. For example, what is the nearest 6 cities with population $>$ 1 million? Without an incremental algorithm, alternatives would be to either repeatedly evaluate for values of k>6 until 6 millionic cities are reported. Another alternative is to scan for millionic cities, rendering the access method (an R-tree) useless, and then build an index and evaluate kNN for k=6. Both are inefficient. Many variations of kNN has since been developed. Visible kNN [17] handles the case where obstacles exist. Reverse-kNN [18] answers the query: find objects where the given point is among the k neighbours of such objects. E.g., find all homes that a proposed gas station location is among the closest 3 gas station to such homes? All nearest neighbours (Ann) [19] is a form of join operation: for all objects get kNN?

2.3 Space Filling Curves

Space Filling curves ($SFCs$) are a linearization of d-dimensional spaces that gives every point in space a scalar value. $SFCs$, similar to fractals, are recursively defined. Initially the curve is defined over unit space and touches ever point, then, it is recursively applied until it reaches the lowest resolution of the space. Peano [20], Hilbert [21], and Morton (Z-Order) [22] are common $SFCs$. Since SFC is a dimensionality reduction, loss of distribution properties happens. Hilbert curve preserves spatial locality (that is if 2 points are close in the un-mapped space; they are found to have Hilbert values that are close on the linear mapping). [23] studied clustering properties of Hilbert curve. [24], [25], [26],and [27] studied further properties of $SFCs$.

2.4 Big Data processing

Implementations of MapReduce [28] [4] proved to be important for processing massive data. A plethora of supporting tools [29] [30] lead to greater adoption. While originally targeted for processing of static data, extensions for updatable key-value stores like Bigtable [31], and iterative processing like Spark [32] emerged. [7] Characterizes that disk locality in data-center computing will no longer be relevant as local RAM will be large enough to host data for processing and is a few orders of magnitude faster than disk and network.

2.5 Big Spatial Data

Several systems targeted adding spatial/multidimensional awareness to big data processing. MD-HBASE [3] utilized Z-Ordering for it's locality preserving properties along with region encoding properties to index multidimensional data in HBASE. Spatial Hadoop [2] utilizes Hilbert curve to build a histogram over a sample of the data and then spatially partitions the data set. CG_Hadoop [33] defines several computational geometry operations on Spatial Hadoop.

3 BALANCED COARSE DISTANCE BROWSING

3.1 Preliminaries

3.1.1 Distance metrics

[1] Introduced Nearest neighbor queries along with an efficient branch and bound algorithm for evaluation of such queries on R Trees. Reducing objects to their minimum bounding rectangles (MBRs). [1] defined the below metrics between an object and a focal point of a kNN query.

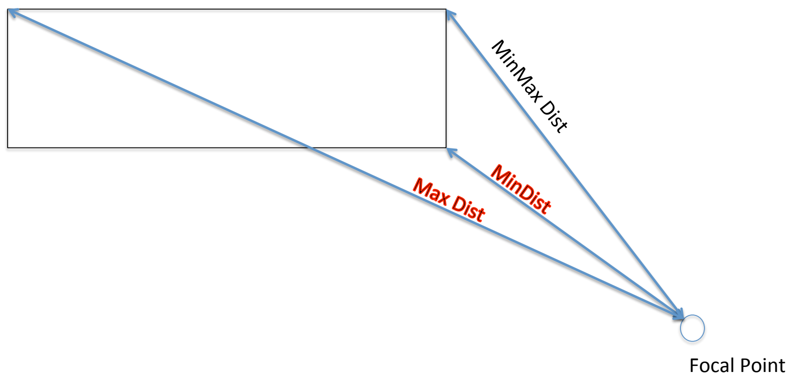


Figure 3.1.: Distance metrics employed by NN search

Minimum distance (MINDIST) is zero if the focal point lies inside the object's MBR otherwise it's the euclidean distance to the nearest edge of the MBR. This is an optimistic metric of containment of an actual object. At this distance an actual object (compared to the bounding geometry) may not exist.

Minimax (MINMAXDIST) To establish a tight lower bound that would guarantee partial containment of an actual object (not just an empty region of it's minimum bounding geometry) [1] defines MINMAXDIST as the euclidean distance to the farthest point on the nearest face of a bounding geometry.

3.1.2 Distance browsing

[5] introduced Distance browsing to incrementally answer kNN queries. This is useful for kNN queries that have an additional relational attribute. For example: what are the 5 nearest cities with population of 5 million? Without incremental evaluation, an approach is to repeatedly evaluate kNN for values ≥ 5 and then test for population. Another approach is to scan for millionic cities (deeming existing R-tree useless) then index the result then apply kNN. Both are inefficient.

3.2 Coarse Distance Browsing

3.2.1 Input Prepossessing

We assume input data is declustered and we have a shared coarse index (R-tree). Declustering can be done Similar to spatial partitioning [2], but instead of bucketing proximal point they are scattered (declusterred) over the partitions. Leaves of the index represent minimum bounding rectangles that are enhanced with count representing how many points in the whole point space reside in such MBR. The index is identical across all partitions.

3.2.2 Evaluating kNN over declustered spatial data

We incrementally consume the coarse index in 2 phases.

phase1 Based on MINDIST metric, we consume the index until k is passed. Until k is reached any uncertain minimum bounding rectangle with MAXDIST smaller than MINDIST of another uncertain MBR is considered certain. This phase maximizes the certain set. After reaching k, for correctness, we add MBRs that overlap with MAXDIST of the uncertain set.

phase2 Based on MAXDIST metric we re-consume the index until k is passed. this is a tighter bound on the uncertain set. After reaching k, for correctness, we add any rooted (of MINDIST smaller than MAXDIST of the uncertain set) to the uncertain set.

An implementation of distance browsing that incrementally reports next proximal Minimum Bounding Rectangle is shown in algorithm 1. Algorithm 2 uses algorithm 1 to get a set of MBRs that are equidistant to the focal point. Algorithm 3 shows phase 1 and 2.

```
Algorithm 1: ExtractMinLeaf Retrieves most proximal leaf MBR.
  Input: Query focal point
  Output: The next leaf MBR of smallest min-dist to query point
1 Assumption  non leaf nodes precedes leaf nodes of similar distance.
2 Assumption  $internalQ \leftarrow newPriorityQeue()$
3 $element \leftarrow internalQ.extractMin()$
4 labelalgo:first if element is non-LEAF tree-node then
5     for ChildNode ∈ element do
6         $internalQ.Enqueue(ChildNode, Distance(ChildNode, QueryPoint))$
7     return ExtractMinLeaf()
8 else if element is LEAF MBR then
9     return element
```

```
Algorithm 2: ExtractMinSet Retrieves next proximal set of equidistant leaf MBRs.
  Input: Query point
  Output: The next set of leaf MBRs of smallest min-dist to query point
1 $crustMinSet.add(ExtractMinLeaf())$
2 while PeekMinLeaf() and crustMinSet are equidistant do
3     $crustMinSet.add(ExtractMinLeaf())$
4 return crustMinSet
```

Algorithm 3: INSITU-NN Accurate reporting of contributing & uncertain elements of k_{nn} in declustered setting.

```
Input: Aligned R-tree, Query point: $p$, Required nearest neighbors:$k$
Output: accurate classification of local elements: Contributing, Pruned, and Uncertain
1  $uncertainSet, contributingSet \leftarrow newSet()$
2  $spaceMinQ, uncertainMaxDistMinQ \leftarrow newMinPriorityQeue()$
3  $spaceMinQ.enqueue(R - tree.Root(), 0)$
4  crustMinSet $\leftarrow \phi$
   // Contributing enlargement phase
5  while spaceMinQ.isNotEmpty() and $size(uncertainMaxDistMinQ \cup contributingSet) < k$ do
6      crustMinSet $\leftarrow spaceMinQ.modifiedExtractMin()$
7      while crustMinSet.minValue > uncertainMaxDistMinQ.peek().maxValue do
8          contributingSet.append(uncertainMaxDistMinQ.extractMin())
9      uncertainMaxDistMinQ.insert(crustMinSet)
10 $contributigLimit \leftarrow MaxDist(contributingSet, p)$
   // Uncertain shrinking phase
11 while $spaceMinQ.peek().minValue \leq uncertainMaxDistMinQ.peek().maxValue$ do
12     uncertainMaxDistMinQ.enqueue(spaceMinQ.modifiedExtractMin())
13 while uncertainMaxDistMinQ..isNotEmpty() and $size(uncertainSet \cup contributingSet) \leq k$ do
14     $uncertainSet.append(uncertainMaxDistMinQ.extractMin())$
15 $uncertainLimit \leftarrow MaxDist(uncertainSet, p)$
   // Adding rooted MBRs( have lower mindist) despite reaching $k$
16 while uncertainMaxDistMinQ.isNotEmpty() do
17     if $uncertainMaxDistMinQ.peek().minValue < uncertainLimit$ then
18         $uncertainSet.append(uncertainMaxDistMinQ.extractMin())$
19 Scan Uncertain MBRs marking all belonging local objects that are farther than (uncertainLimit)
     as Non-Contributing.
20 Scan Uncertain MBRs marking their local objects that are Nearer than (contributigLimit) as
     Contributing.
21 Report Contributing & Uncertain concrete objects, and Non-Contributing
     boundary(Farthest(lastUncertainUncertain))
```

4 ANALYSIS

Section 4.1 derives an analytic estimate for uncertainty. Section 4.2 empirically measures uncertainty for various data sets. Section 4.3 makes concluding remarks.

4.1 Analytic Estimate of Uncertainty for a uniform data set

4.1.1 Sources of uncertainty

A representative source of uncertainty is when 2 minimum bounding rectangles have only partial (not total) ordering between them w.r.t proximity to the focal point, i.e., neither is wholly more proximate than the other to the focal point (see figure 4.1).

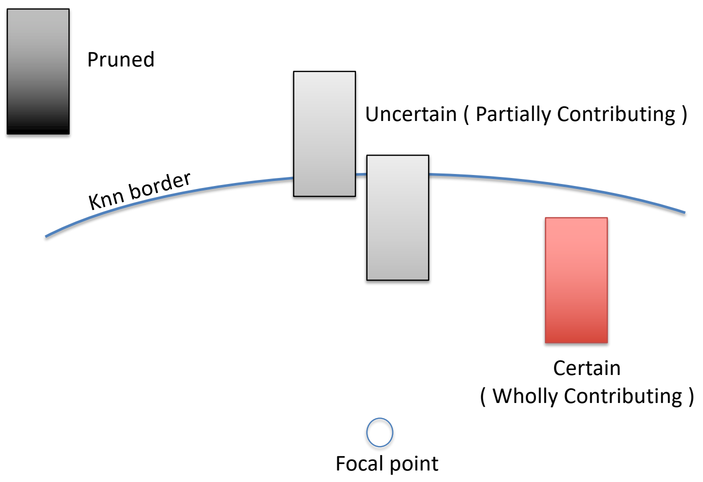


Figure 4.1.: Sources of Uncertainty: For a kNN operation over a uniform point space: The perimeter of the k nearest neighbouring points is a circle around the focal point (the kNN border). If the kNN operation were to be done in a coarse fashion (i.e., over leaf level MBR, not the points) uncertainty would arise as some MBR (in grey) can only have partial ordering w.r.t each other. I.e., neither is wholly closer to the focal point. Black represents a pruned MBR.

4.1.2 Proof Sketch

We assume 1) a uniform point space, 2) the granularity of calculating kNN is restricted to leaf level minimum bounding rectangles. We relax point count to areas and compare areas of an exact kNN to area of uncertain sets then derive a relation ship in terms of k.

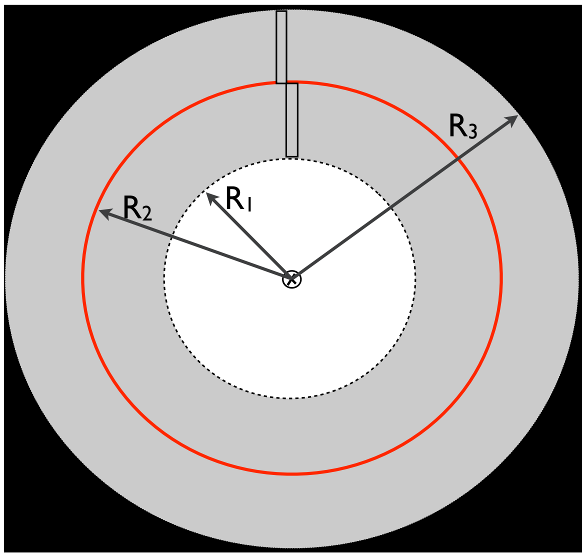


Figure 4.2.: Areas of pruned, certain, and uncertain sets: Circle of R2 represents the exact kNN border. Circle of R1 represents the outer bound of leaf MBRs that are wholly closer to the focal point and $\leq k$. Circle of R3 represents outer bound of leaf MBRs that are are uncertain.

$$\frac{Area_{uncertain}}{Area_{k_{nn}}} = \frac{\pi(r_3^2 - r_1^2)}{\pi r_2^2} < \frac{r_3^2 - r_1^2}{r_1^2} \tag{4.1}$$

$$r_3 = r_1 + 2 * length_{mbr} \tag{4.2}$$

$$\frac{Area_{uncertain}}{Area_{k_{nn}}} < \frac{(r_1 + 2 * length_{mbr})^2 - r_1^2}{r_1^2} \tag{4.3}$$

$$\frac{Area_{uncertain}}{Area_{k_{nn}}} < \frac{\cancel{r_1^2} + (4 * r_1 - length_{mbr}) * length_{mbr} - \cancel{r_1^2}}{r_1^2} \tag{4.4}$$

$$Avg(Area_{MBR}) = m/d \text{ where } \begin{cases} m \text{ is the R-tree branching factor} \\ d \text{ is point space's density} \\ m\&d \text{ are both constants} \end{cases} \tag{4.5}$$

$$length_{mbr} < m/d \mid \text{assuming unit or fixed } width_{mbr}. \tag{4.6}$$

$$\frac{Area_{uncertain}}{Area_{k_{nn}}} < \frac{1}{r_1} \tag{4.7}$$

$$r_1 \propto \sqrt{k} \tag{4.8}$$

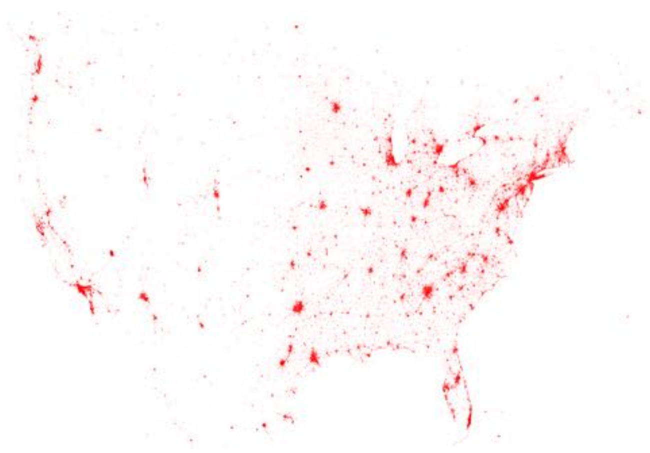
$$\frac{Area_{uncertain}}{Area_{k_{nn}}} < \frac{1}{\sqrt{k}} \tag{4.9}$$

4.2 Empirical Analysis

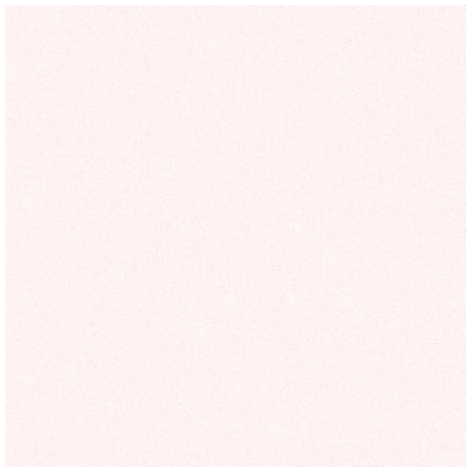
4.2.1 Data Sets

We used 4 point data sets as shown in figure 4.3. Visualization of corresponding leaf level minimum bounding rectangles is shown in figure 4.4.

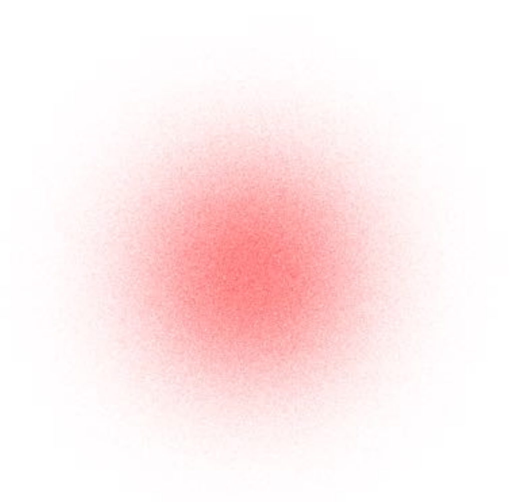
- Synthetic: Uniform, Correlated, Gaussian
- Twitter Data: U.S. 2013 tweets location



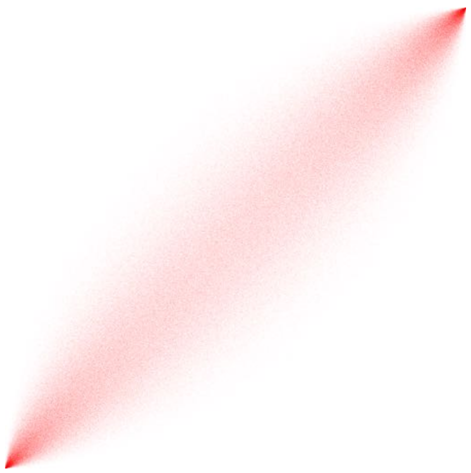
(1) US-2013-Tweet locations



(2) Uniform

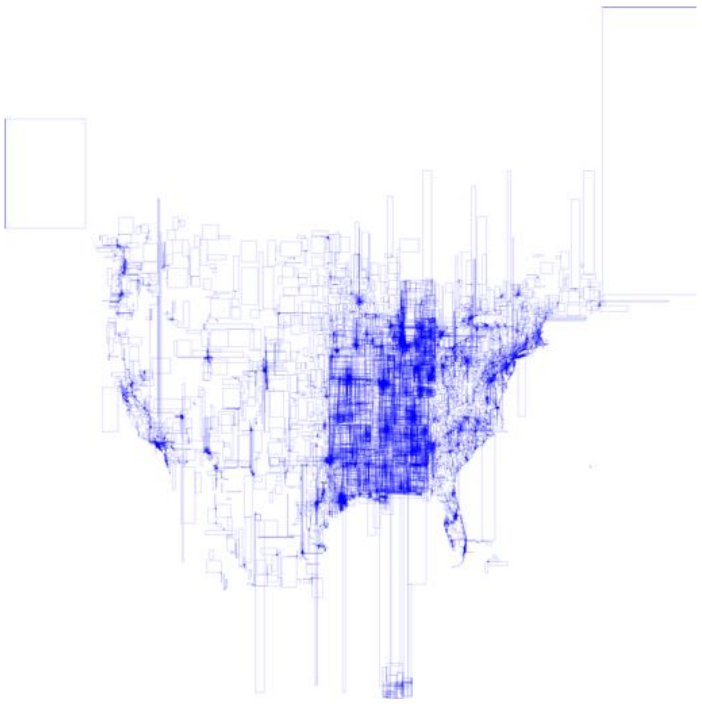


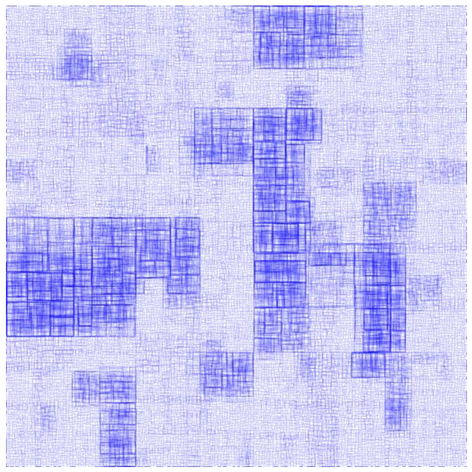
(3) Gaussian

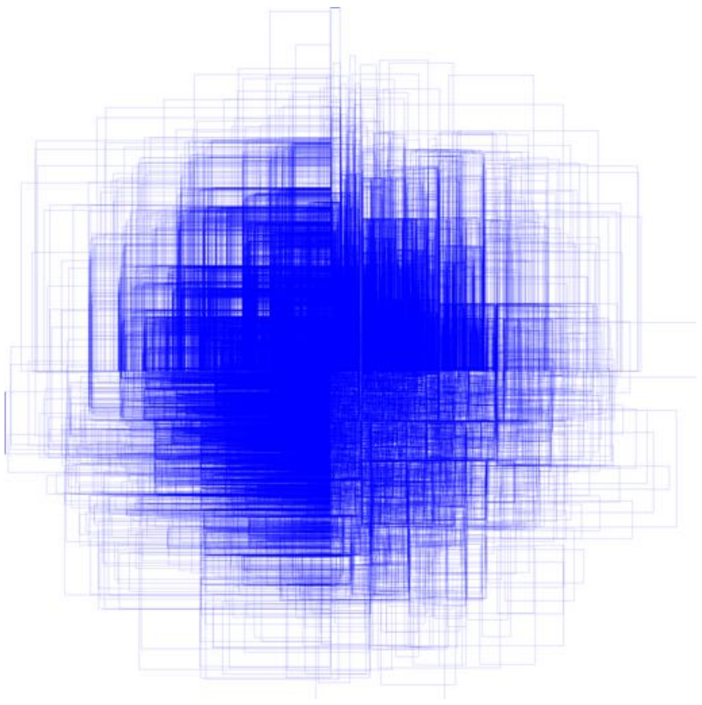


(4) Correlated

Figure 4.3.: Point data sets


(1) US-2013-Tweet locations


(2) Uniform


(3) Gaussian

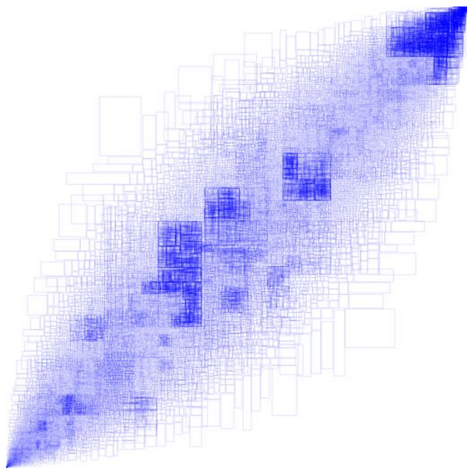
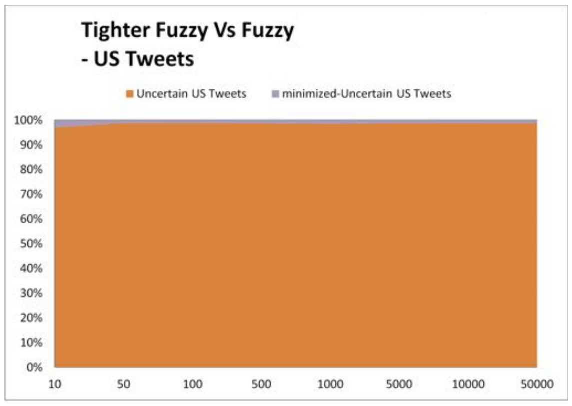

(4) Correlated

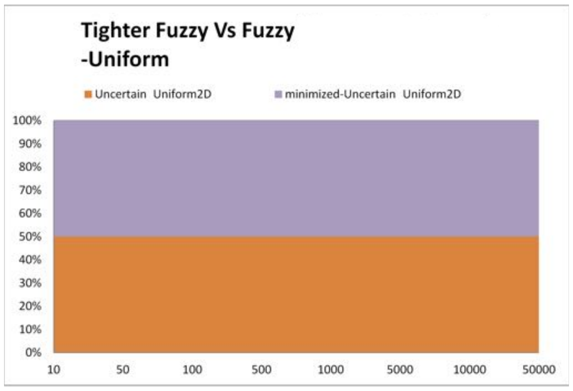
Figure 4.4.: Leaf level minimum bounding rectangles of an R-tree populated with various data sets

4.2.2 Savings of phase 2

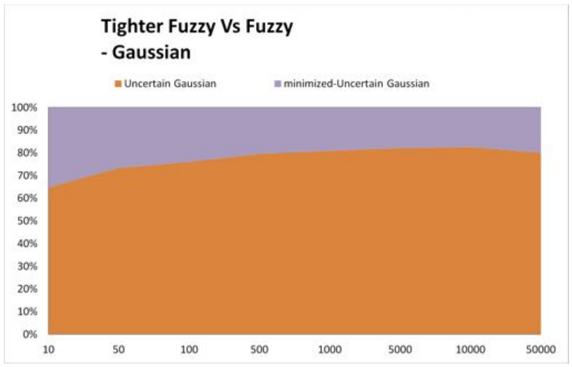
We measure how tight phase 2 of the algorithm is compared to phase 1 for various values of k.



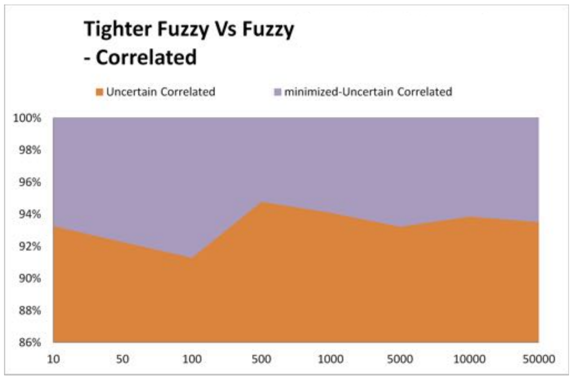
(1) US-2013-Tweet locations



(2) Uniform



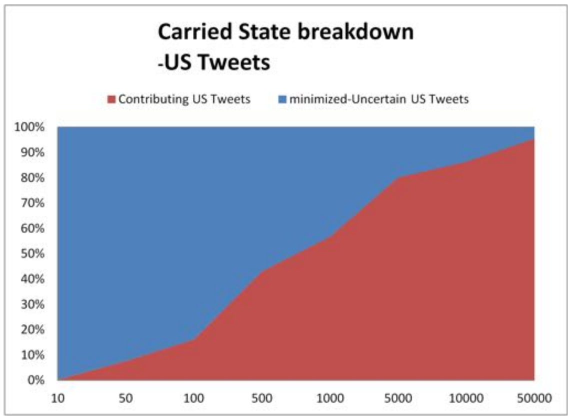
(3) Gaussian



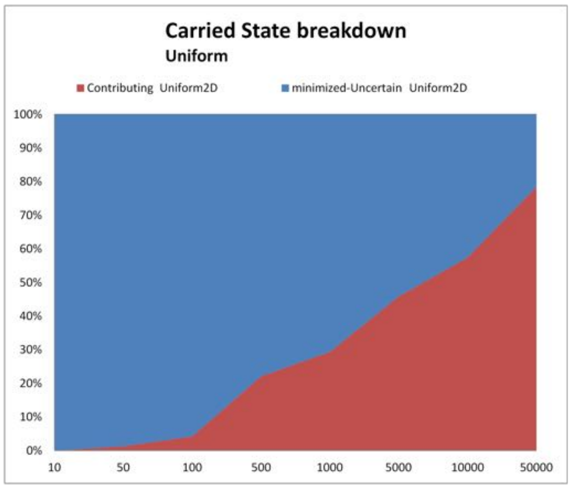
(4) Correlated

Figure 4.5.: Tight Uncertain set of phase 2 vs Uncertain set of phase 1 for various values of k. More disparity is better

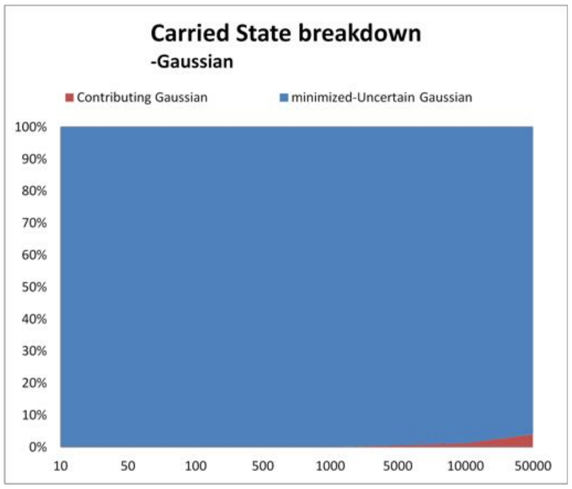
4.2.3 Certain set vs Uncertain set



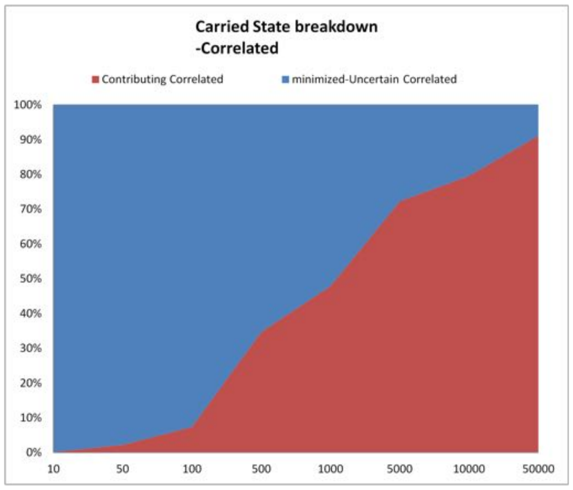
(1) US-2013-Tweet locations



(2) Uniform



(3) Gaussian



(4) Correlated

Figure 4.6.: Break down of the carried state: Certain vs Uncertain

4.2.4 Empirical Estimate of Uncertainty

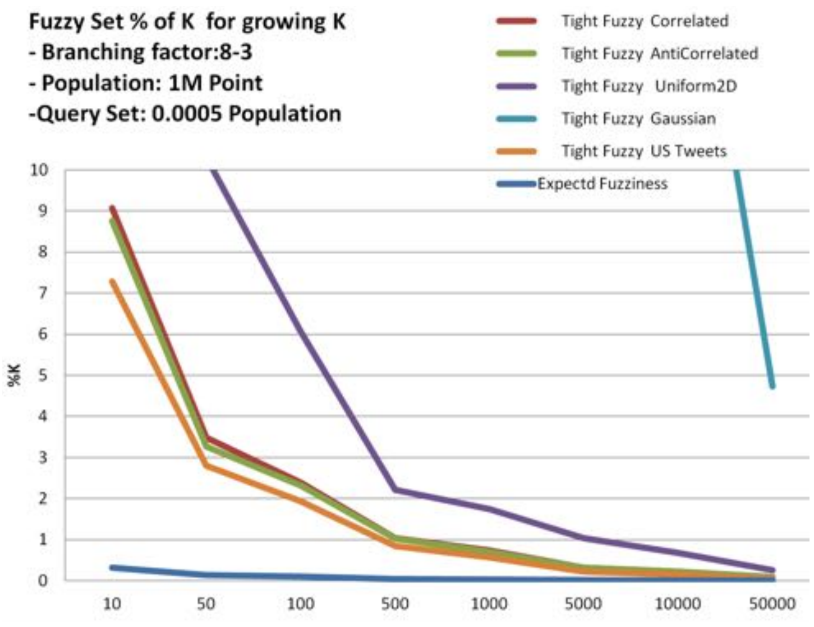


Figure 4.7.: Empirical estimate of uncertainty: x-axis is various values of k, y-axis is size of uncertain set in multiples of k

4.2.5 Certain set size vs k

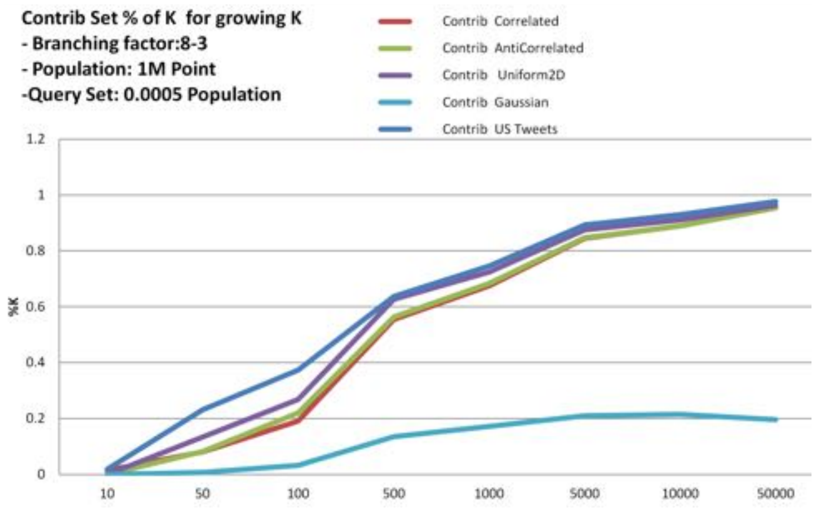


Figure 4.8.: Certain set size in terms of multiples of k for various values of k

4.3 Summary

Below is a summary of findings

- Analytically, for uniform data set, uncertainty is O($\sqrt{k}$)
- Empirically, all of the data sets showed higher uncertainty than analytically derived. The worst is the Gaussian distribution (see figures 4.6 and 4.7).
- Empirically, for all distributions, uncertainty is smaller for larger values of k (see figure 4.7).
- Empirically, size of certain set increases for larger values of k (see figure 4.8).
- Empirically, savings of phase 2 were most for real life data: US 2013 Tweets (see figure 4.5).

4.4 Future work

- Exploring and deriving a bound on uncertainty of partial evaluation is useful for query optimization. For example,**Pushing of spatial selects** over partially evaluated kNN: if a select operator were to consume the output of a kNN query: full evaluation of kNN is needed. If partial evaluation of kNN has non trivial bounds on uncertainty (and certainty) it can evaluate a spatial select as it could lie entirely in the pruned region or the certain region eliminating the need to resolve the uncertainty.
- Studying more operators and deriving bounds on uncertainty in absence of communication, then defining equivalence relations among possible operator compositions that can be useful for a cost based query optimizer.

REFERENCES

[1] Nick Roussopoulos, Stephen Kelley, and Frédéric Vincent. Nearest neighbor queries. In *ACM sigmod record*, volume 24, pages 71–79. ACM, 1995.

[2] Ahmed Eldawy and Mohamed F Mokbel. Spatialhadoop: A mapreduce framework for spatial data. In *2015 IEEE 31st international conference on Data Engineering*, pages 1352–1363. IEEE, 2015.

[3] Shoji Nishimura, Sudipto Das, Divyakant Agrawal, and Amr El Abbadi. Md-hbase: a scalable multi-dimensional data infrastructure for location aware services. In *Mobile Data Management (MDM), 2011 12th IEEE International Conference on*, volume 1, pages 7–16. IEEE, 2011.

[4] Ablimit Aji, Fusheng Wang, Hoang Vo, Rubao Lee, Qiaoling Liu, Xiaodong Zhang, and Joel Saltz. Hadoop gis: a high performance spatial data warehousing system over mapreduce. *Proceedings of the VLDB Endowment*, 6(11):1009–1020, 2013.

[5] Gísli R Hjaltason and Hanan Samet. Distance browsing in spatial databases. *ACM Transactions on Database Systems (TODS)*, 24(2):265–318, 1999.

[6] Matthias Felleisen, Robert Bruce Findler, Matthew Flatt, and Shriram Krishnamurthi. How to design programs, 2001.

[7] Ganesh Ananthanarayanan, Ali Ghodsi, Scott Shenker, and Ion Stoica. Disk-locality in datacenter computing considered irrelevant. In *Proceedings of the 13th USENIX Conference on Hot Topics in Operating Systems*, HotOS'13, pages 12–12, Berkeley, CA, USA, 2011. USENIX Association.

[8] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. *SIGMOD Rec.*, 14(2):47–57, June 1984.

[9] Douglas Comer. Ubiquitous b-tree. *ACM Comput. Surv.*, 11(2):121–137, June 1979.

[10] Timos Sellis, Nick Roussopoulos, and Christos Faloutsos. The r+-tree: A dynamic index for multi-dimensional objects. pages 507–518, 1987.

[11] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The r*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, SIGMOD '90, pages 322–331, New York, NY, USA, 1990. ACM.

[12] Ibrahim Kamel and Christos Faloutsos. Hilbert r-tree: An improved r-tree using fractals. In *VLDB*, 1994.

[13] Nick Roussopoulos and Daniel Leifker. Direct spatial search on pictorial databases using packed r-trees. *ACM Sigmod Record*, 14(4):17–31, 1985.

[14] Ibrahim Kamel and Christos Faloutsos. On packing r-trees. In *CIKM*, 1993.

[15] Scott T Leutenegger, Mario A Lopez, and Jeffrey Edgington. Str: A simple and efficient algorithm for r-tree packing. In *Proceedings 13th International Conference on Data Engineering*, pages 497–506. IEEE, 1997.

[16] Yannis Manolopoulos, Alexandros Nanopoulos, Apostolos N Papadopoulos, and Yannis Theodoridis. *R-trees: Theory and Applications.* Springer Science & Business Media, 2010.

[17] Sarana Nutanong, Egemen Tanin, and Rui Zhang. Visible nearest neighbor queries. In *International Conference on Database Systems for Advanced Applications*, pages 876–883. Springer, 2007.

[18] Flip Korn and Suresh Muthukrishnan. Influence sets based on reverse nearest neighbor queries. *ACM Sigmod Record*, 29(2):201–212, 2000.

[19] Jun Zhang, Nikos Mamoulis, Dimitris Papadias, and Yufei Tao. All-nearest-neighbors queries in spatial databases. In *Proceedings. 16th International Conference on Scientific and Statistical Database Management, 2004.*, pages 297–306. IEEE, 2004.

[20] Giuseppe Peano. Sur une courbe, qui remplit toute une aire plane. *Mathematische Annalen*, 36(1):157–160, 1890.

[21] David Hilbert. Ueber die stetige abbildung einer line auf ein flächenstück. *Mathematische Annalen*, 38(3):459–460, Sep 1891.

[22] Guy M Morton. A computer oriented geodetic data base and a new technique in file sequencing. 1966.

[23] Bongki Moon, Hosagrahar V Jagadish, Christos Faloutsos, and Joel H. Saltz. Analysis of the clustering properties of the hilbert space-filling curve. *IEEE Transactions on knowledge and data engineering*, 13(1):124–141, 2001.

[24] Craig Gotsman and Michael Lindenbaum. On the metric properties of discrete space-filling curves. *IEEE Transactions on Image Processing*, 5(5):794–797, 1996.

[25] Mohamed F Mokbel, Walid G Aref, and Ibrahim Kamel. Performance of multi-dimensional space-filling curves. In *Proceedings of the 10th ACM international symposium on Advances in geographic information systems*, pages 149–154. ACM, 2002.

[26] Mohamed F. Mokbel and Walid G. Aref. Irregularity in multi-dimensional space-filling curves with applications in multimedia databases. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, CIKM '01, pages 512–519, New York, NY, USA, 2001. ACM.

[27] Mohamed F Mokbel, Walid G Aref, and Ibrahim Kamel. Analysis of multi-dimensional space-filling curves. *GeoInformatica*, 7(3):179–209, 2003.

[28] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[29] Christopher Olston, Benjamin Reed, Utkarsh Srivastava, Ravi Kumar, and Andrew Tomkins. Pig latin: a not-so-foreign language for data processing. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1099–1110. ACM, 2008.

[30] Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Ning Zhang, Suresh Antony, Hao Liu, and Raghotham Murthy. Hive-a petabyte scale data warehouse using hadoop. In *2010 IEEE 26th international conference on data engineering (ICDE 2010)*, pages 996–1005. IEEE, 2010.

[31] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C Hsieh, Deborah A Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E Gruber. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2):4, 2008.

[32] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. *HotCloud*, 10(10-10):95, 2010.

[33] Ahmed Eldawy, Yuan Li, Mohamed F Mokbel, and Ravi Janardan. Cg_hadoop: computational geometry in mapreduce. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 294–303. ACM, 2013.

A VISUALIZED EXECUTION TRACE OF COARSE DISTANCE BROWSING

Following, we graphically show a step by step execution trace of coarse distance browsing to evaluate kNN instance over uniform data set. For this instance; k=100. In each step, leaf level minimum bounding rectangles are highlighted into three groups: Certain (wholly contributing), Uncertain (partially contributing), Pruned. Count of enclosed points is written on top left of each leaf level MBR, and on top of perimeters if any. The trace is shown in two phases: appendix A.1 for **Certain set** enlargement phase, and appendix A.2 for **Uncertain set** minimization phase.

A.1 **Certain set** enlargement phase

In this phase we start with an empty uncertain and certain sets. We expand the **uncertain set** based on MINDIST until sum of both sets exceeds k. While (uncertain set + certain set $< k$) and the **next proximal** (based on MINDIST) MBR is fully dominated by uncertain MBR(s) then those dominating MBRs are deemed **certain**. MBR X dominates MBR Y when every point in X is closer to the focal point than any point in Y, i.e., when MAXDIST of X is $\leq$ MINDIST of Y. This is demonstrated graphically in figures A.1 to A.19.

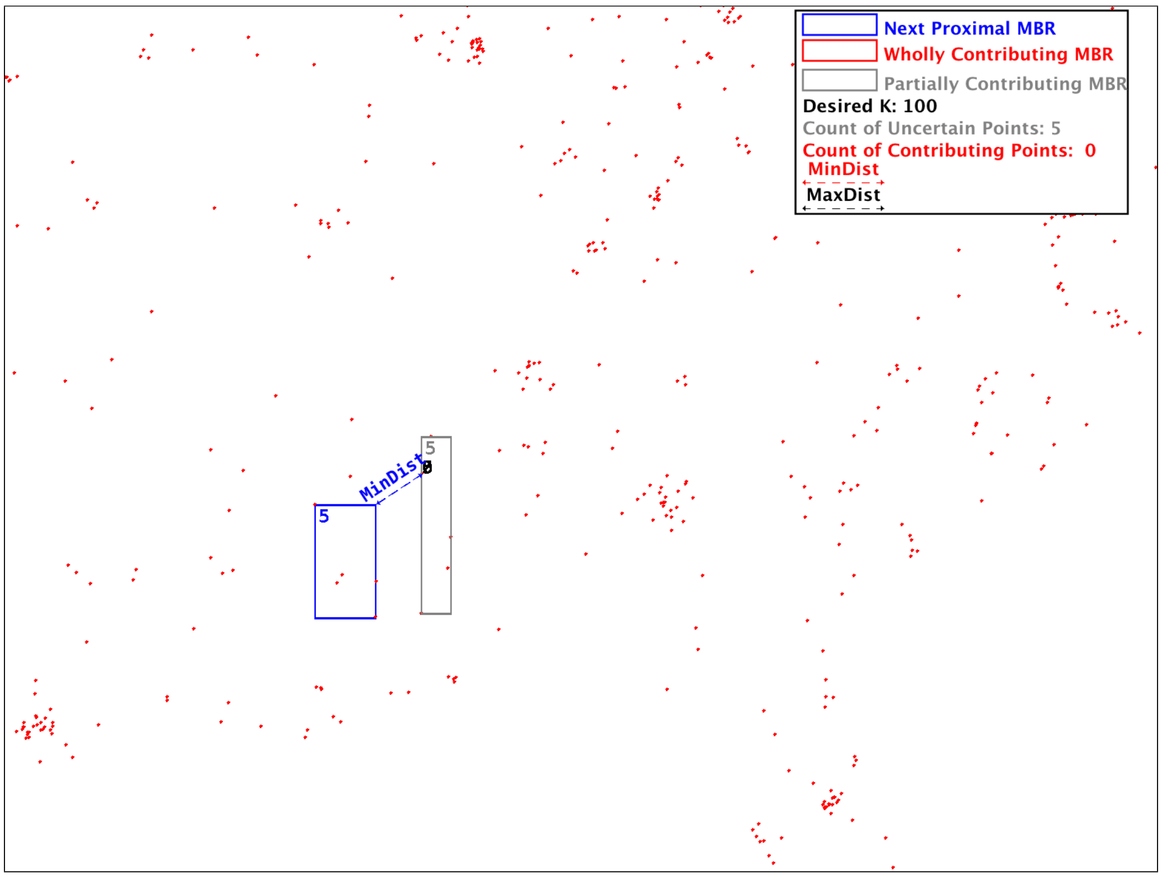

Next Proximal MBR
Wholly Contributing MBR
Partially Contributing MBR
Desired K: 100
Count of Uncertain Points: 5
Count of Contributing Points: 0
MinDist
MaxDist
MinDist
5
5
0

Figure A.1.: **Certain set** enlargement phase step 1: The focal point exists in an MBR (i.e., of 0 MINDIST to the focal point) so that automatically adds the MBR to the uncertain set (in grey). Uncertain set is now totaling 5 points.Since current total points (**certain** + **uncertain** = 5) is below (k = 100), **next proximal MBR**, based on MINDIST, will be added to the uncertain set.

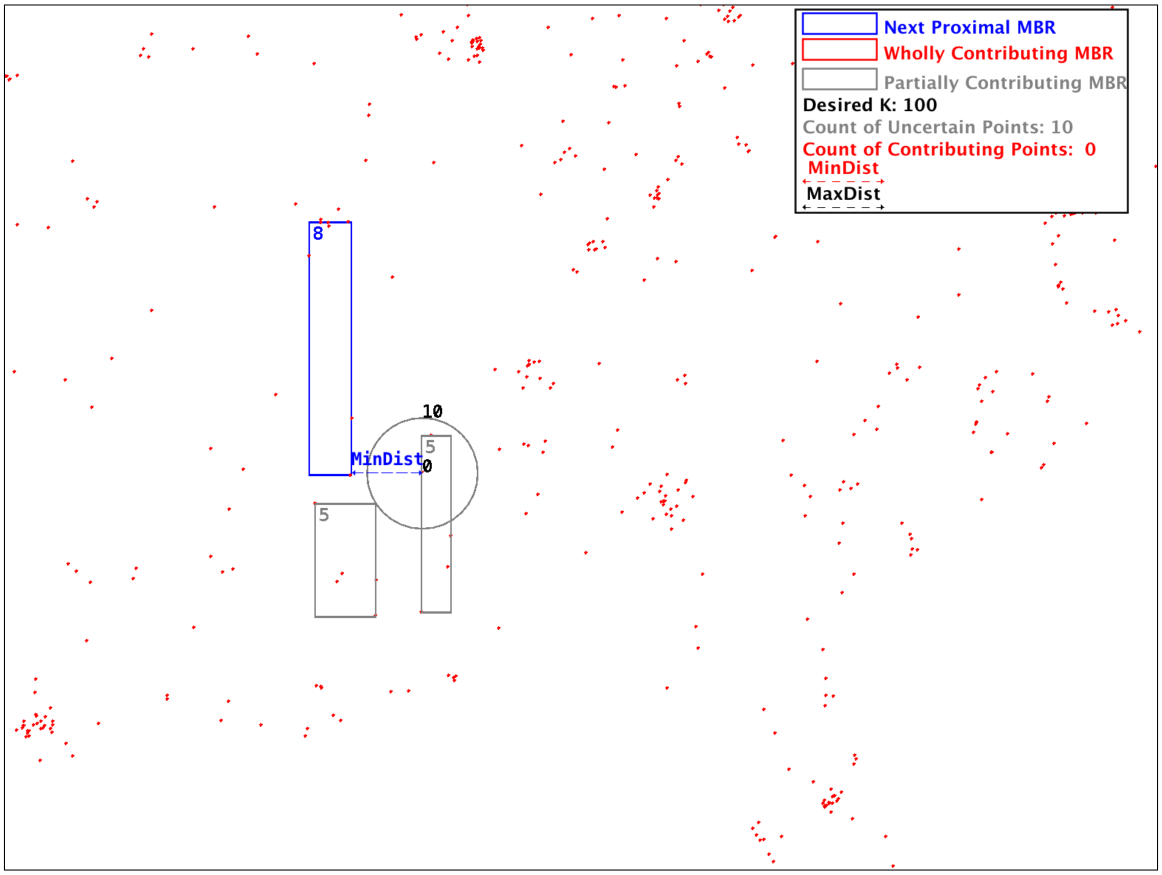

Next Proximal MBR
Wholly Contributing MBR
Partially Contributing MBR
Desired K: 100
Count of Uncertain Points: 10
Count of Contributing Points: 0
MinDist
MaxDist
8
10
5
0
MinDist
5

Figure A.2.: **Certain set** enlargement phase step 2: Since current total points (**certain** + **uncertain** = 10) is below (k = 100), **next proximal MBR**, based on MINDIST, will be added to the uncertain set.

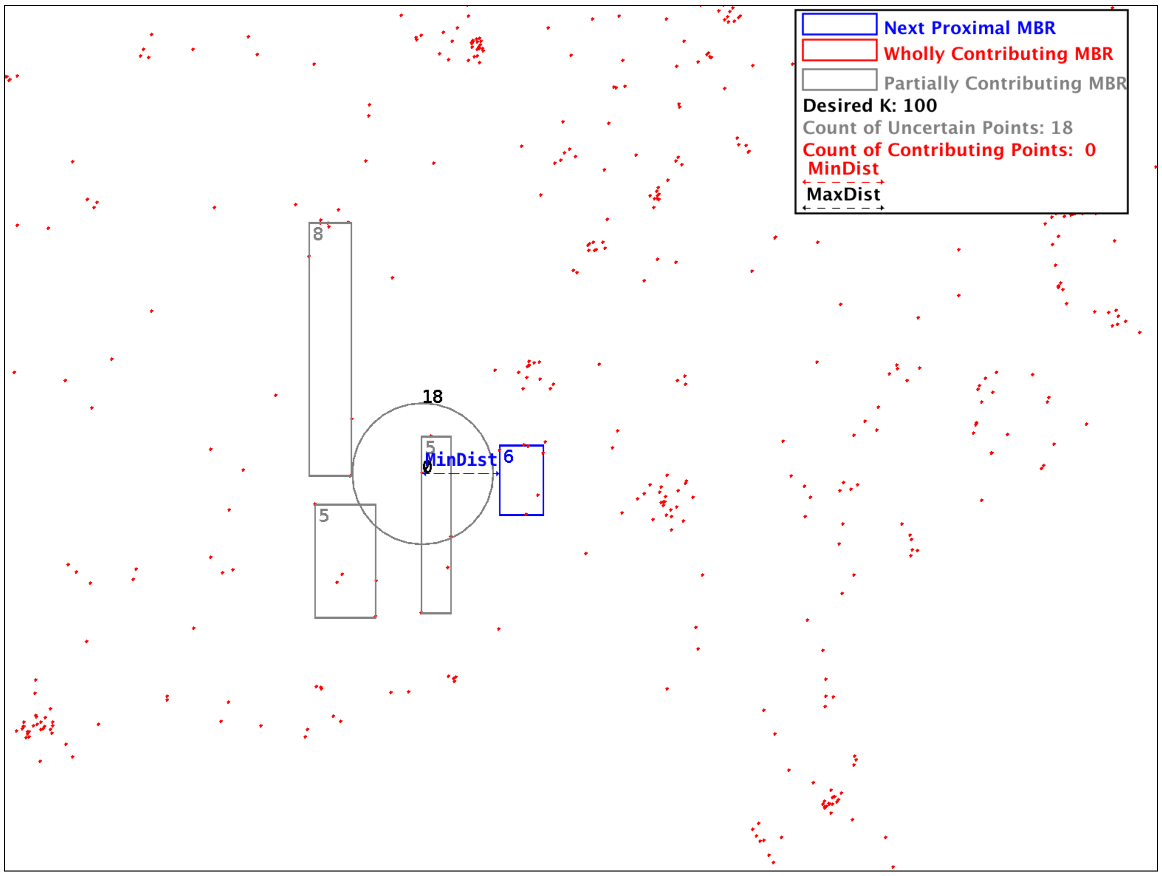

Next Proximal MBR
Wholly Contributing MBR
Partially Contributing MBR
Desired K: 100
Count of Uncertain Points: 18
Count of Contributing Points: 0
MinDist
MaxDist
8
18
5
MinDist
6
0
5

Figure A.3.: **Certain set** enlargement phase step 3: Since current total points (**certain** + **uncertain** = 18) is below (k = 100), **next proximal MBR**, based on MINDIST, will be added to the uncertain set.

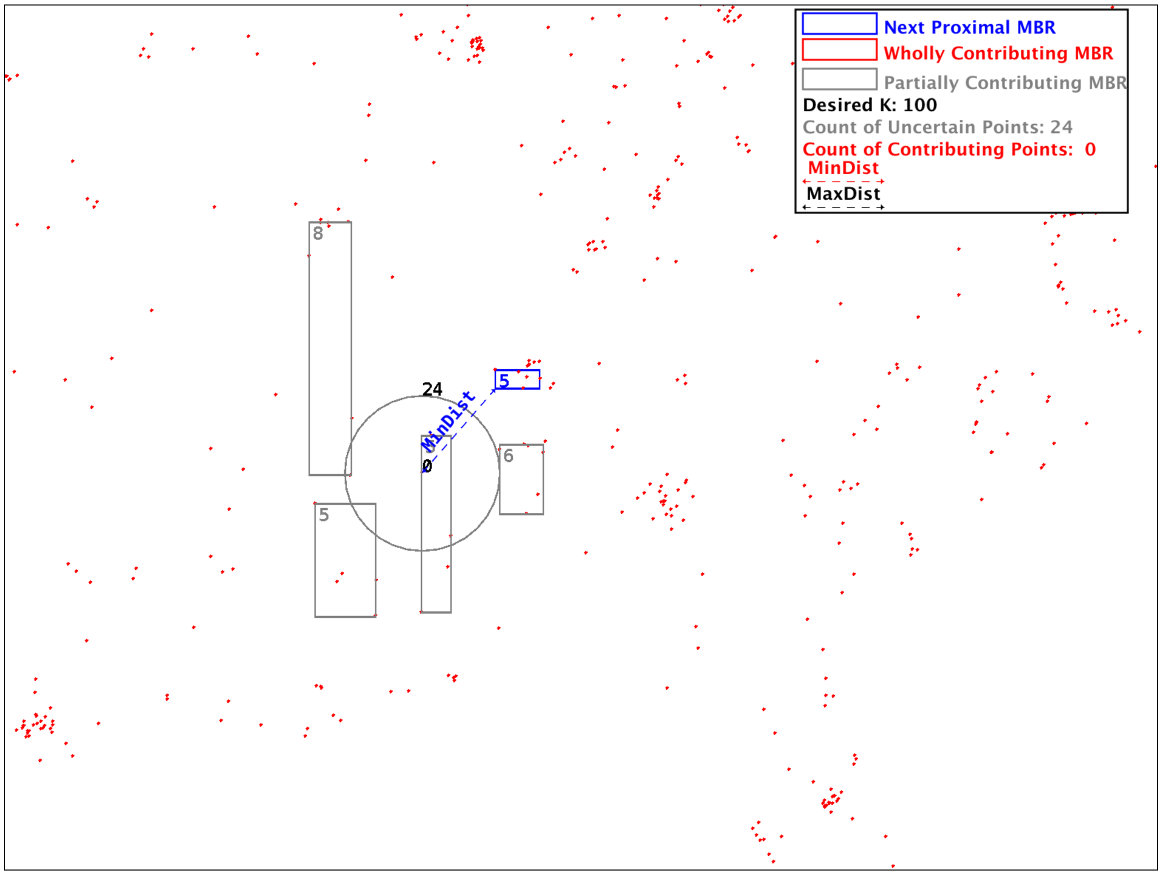

Next Proximal MBR
Wholly Contributing MBR
Partially Contributing MBR
Desired K: 100
Count of Uncertain Points: 24
Count of Contributing Points: 0
MinDist
MaxDist
8
24
MinDist
5
6
0
5

Figure A.4.: **Certain set** enlargement phase step 4: Since current total points (**certain** + **uncertain** = 24) is below (k = 100), **next proximal MBR**, based on MINDIST, will be added to the uncertain set.

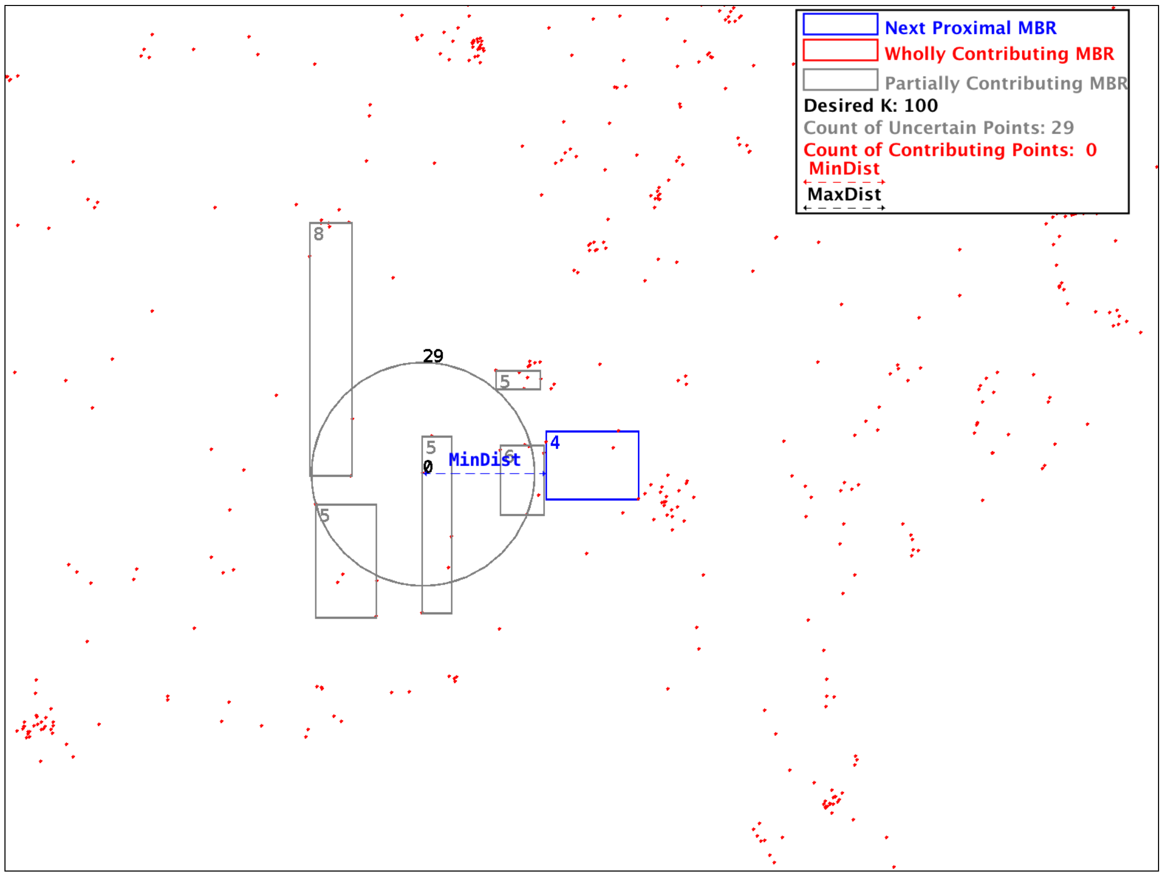

Next Proximal MBR
Wholly Contributing MBR
Partially Contributing MBR
Desired K: 100
Count of Uncertain Points: 29
Count of Contributing Points: 0
MinDist
MaxDist
29
8
5
5
6
4
5
0
MinDist

Figure A.5.: **Certain set** enlargement phase step 5: Since current total points (**certain** + **uncertain** = 29) is below (k = 100), **next proximal MBR**, based on MINDIST, will be added to the uncertain set.

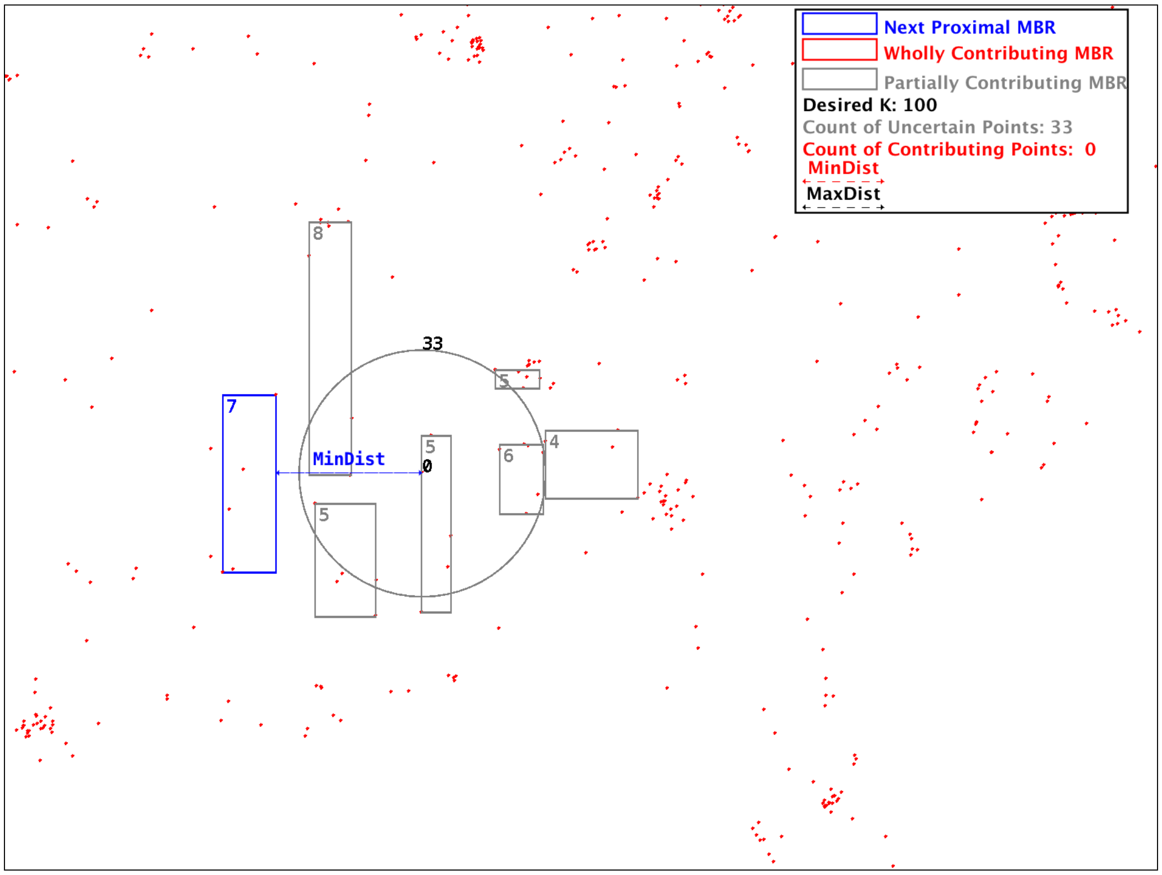

Next Proximal MBR
Wholly Contributing MBR
Partially Contributing MBR
Desired K: 100
Count of Uncertain Points: 33
Count of Contributing Points: 0
MinDist
MaxDist
33
8
7
MinDist
5
0
5
6
4
5

Figure A.6.: **Certain set** enlargement phase step 6: Since current total points (**certain** + **uncertain** = 33) is below (k = 100), **next proximal MBR**, based on MINDIST, will be added to the uncertain set.

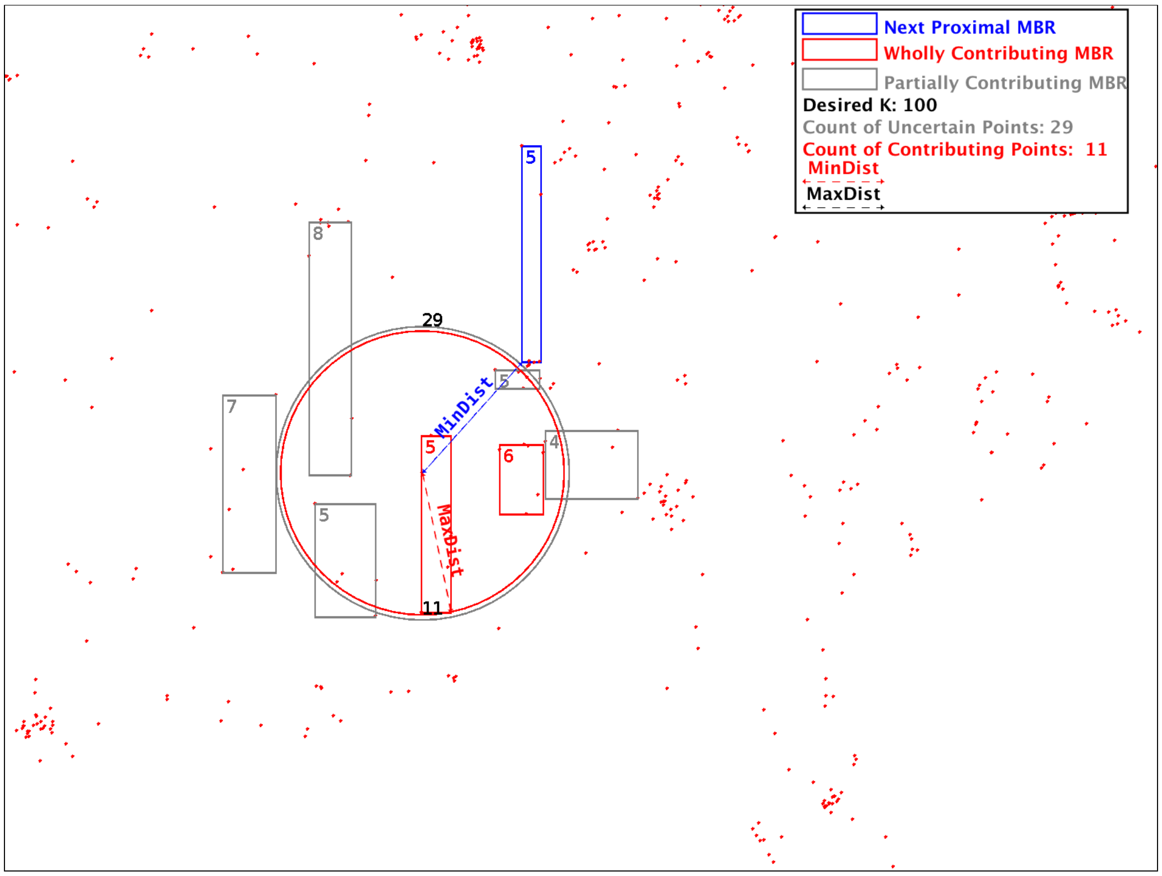


Figure A.7.: **Certain set** enlargement phase step 7: Since current total points (**certain** + **uncertain** = 40) is below (k = 100), **next proximal MBR**, based on MINDIST, will be added to the uncertain set. **Additionally**, since **next proximal MBR** is dominated by **2 uncertain MBRs** and current total is under k, those 2 MBRs are marked as **certain**. MBR X dominates MBR Y when every point in X is closer to the focal point than any point in Y. i.e. more conservatively, when MAXDIST of X is $\leq$ MINDIST of Y.

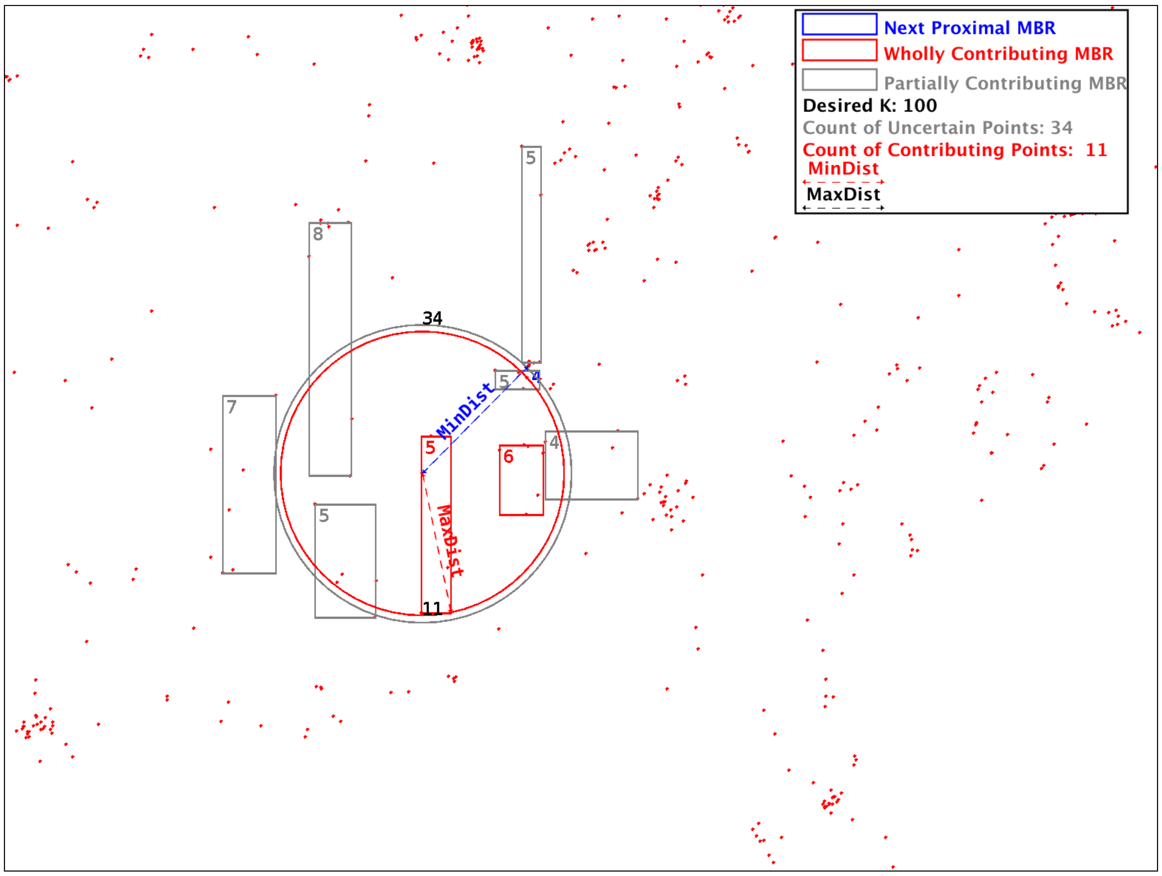


Figure A.8.: **Certain set** enlargement phase step 8: Since current total points (**certain** + **uncertain** = 45) is below (k = 100), **next proximal MBR**, based on MINDIST, will be added to the uncertain set.

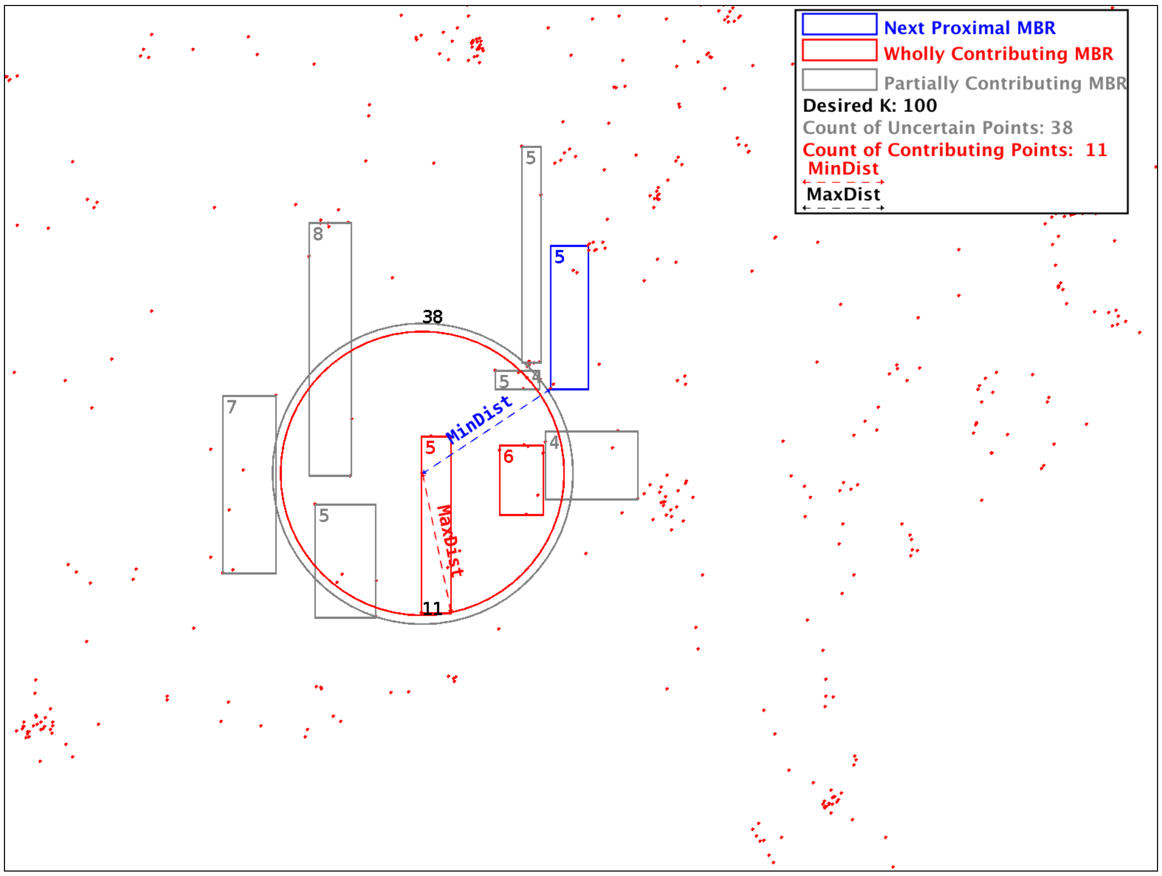


Figure A.9.: **Certain set** enlargement phase step 9: Since current total points (**certain** + **uncertain** = 49) is below (k = 100), **next proximal MBR**, based on MINDIST, will be added to the uncertain set.

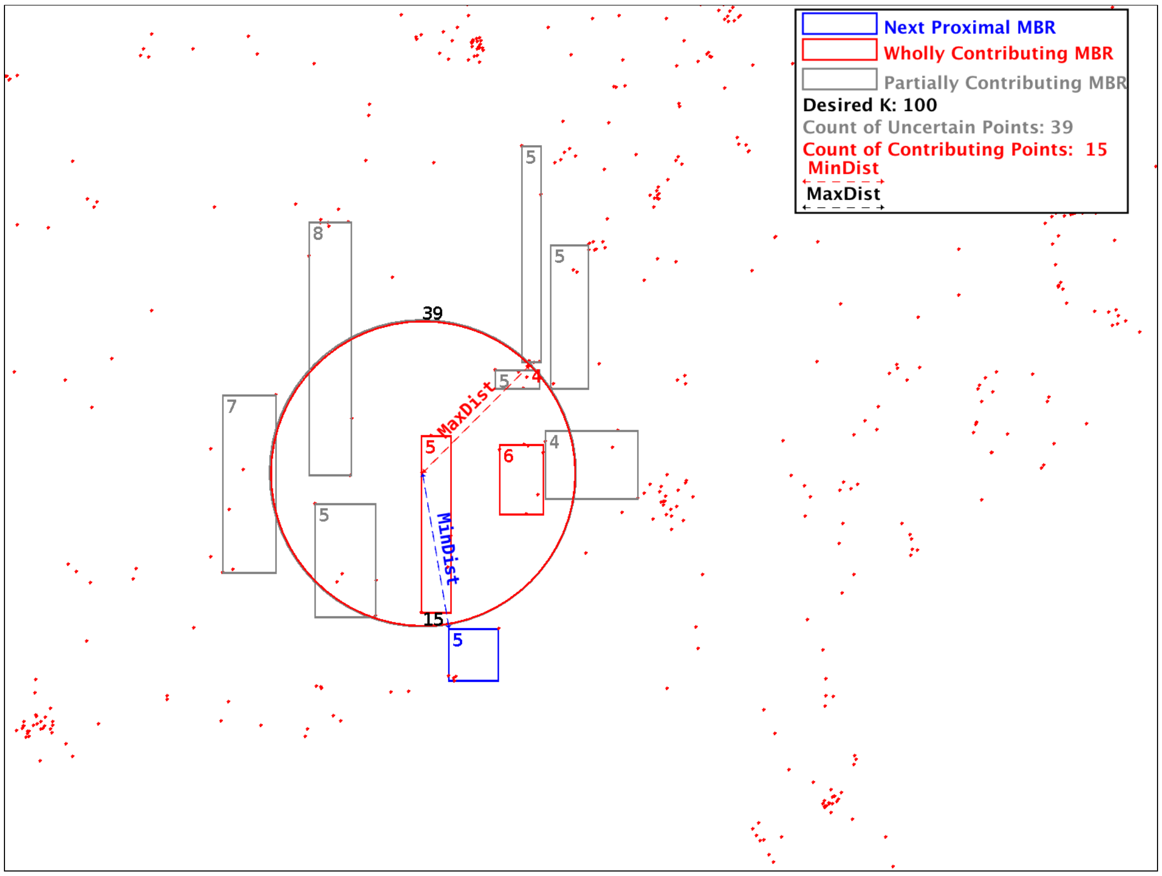


Figure A.10.: **Certain set** enlargement phase step 10: Since current total points (**certain** + **uncertain** = 54) is below (k = 100), **next proximal MBR**, based on MINDIST, will be added to the uncertain set. **Additionally**, since **next proximal MBR** is dominated by an **uncertain MBR (tiny with 4 points)** and current total is under k, this tiny MBR is marked as **certain**. MBR X dominates MBR Y when every point in X is closer to the focal point than any point in Y. i.e. more conservatively, when MAXDIST of X is $\leq$ MINDIST of Y.

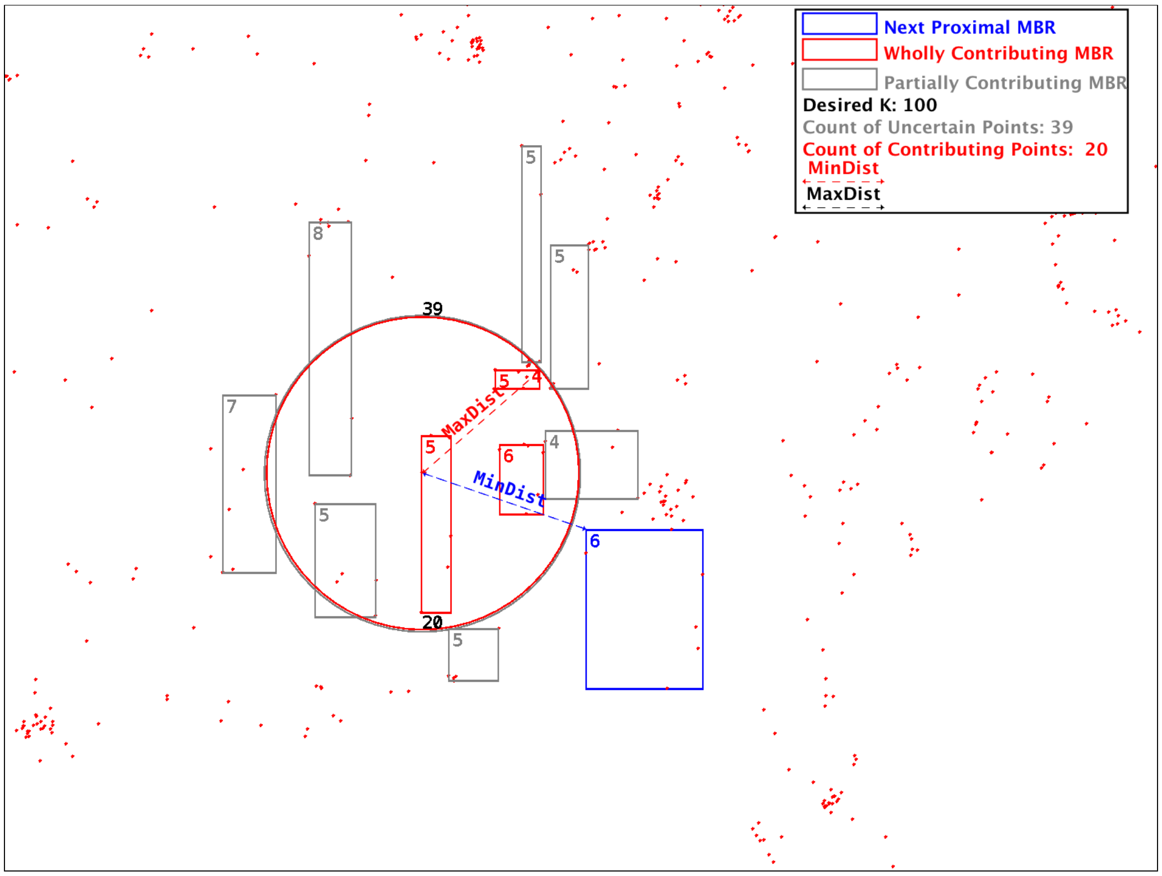


Figure A.11.: **Certain set** enlargement phase step 11: Since current total points (**certain** + **uncertain** = 59) is below (k = 100), **next proximal MBR**, based on MINDIST, will be added to the uncertain set. **Additionally**, since **next proximal MBR** is dominated by an **uncertain MBR** and current total is under k, this MBR is marked as **certain**. MBR X dominates MBR Y when every point in X is closer to the focal point than any point in Y. i.e. more conservatively, when MAXDIST of X is $\leq$ MINDIST of Y.

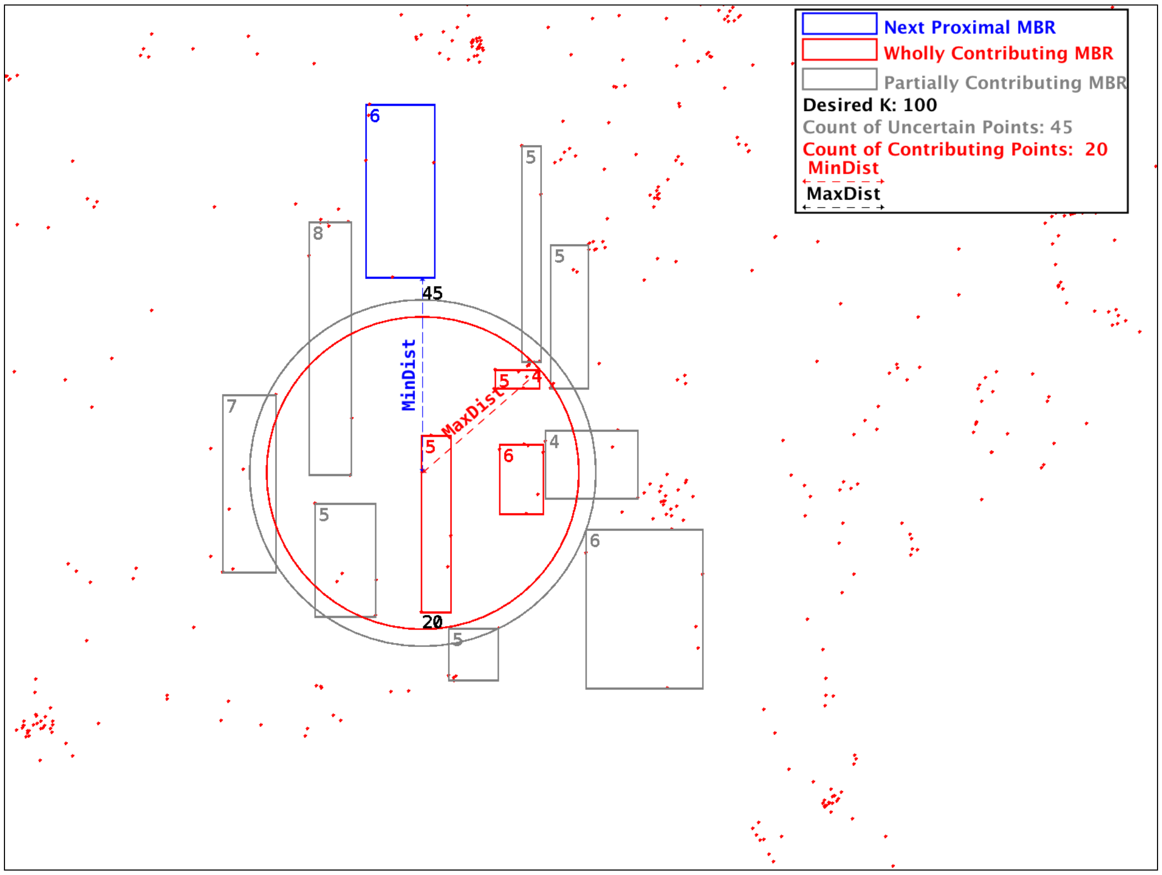


Figure A.12.: **Certain set** enlargement phase step 12: Since current total points (**certain** + **uncertain** = 65) is below (k = 100), **next proximal MBR**, based on MINDIST, will be added to the uncertain set.

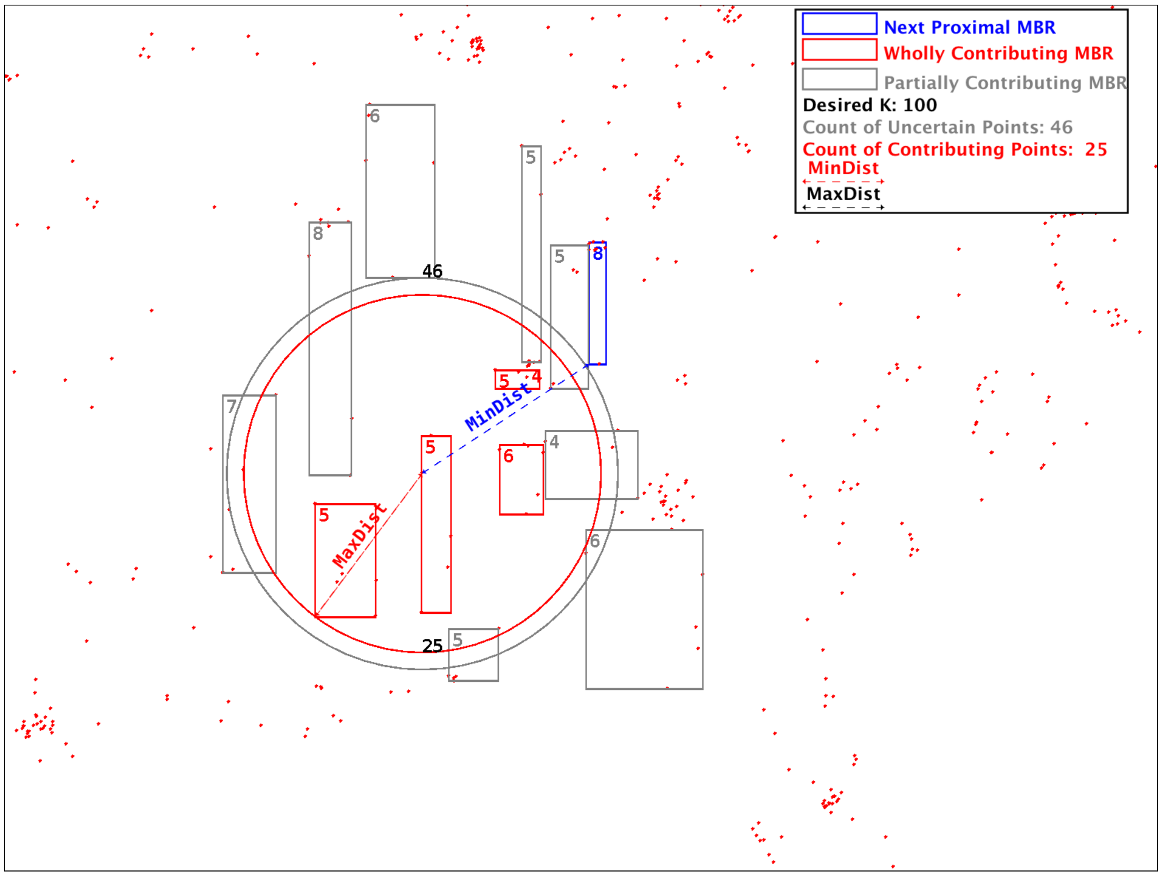


Figure A.13.: **Certain set** enlargement phase step 13: Since current total points (**certain** + **uncertain** = 71) is below (k = 100), **next proximal MBR**, based on MINDIST, will be added to the uncertain set. **Additionally**, since **next proximal MBR** is dominated by an **uncertain MBR** and current total is under k, this MBR is marked as **certain**. MBR X dominates MBR Y when every point in X is closer to the focal point than any point in Y. i.e. more conservatively, when MAXDIST of X is $\leq$ MINDIST of Y.

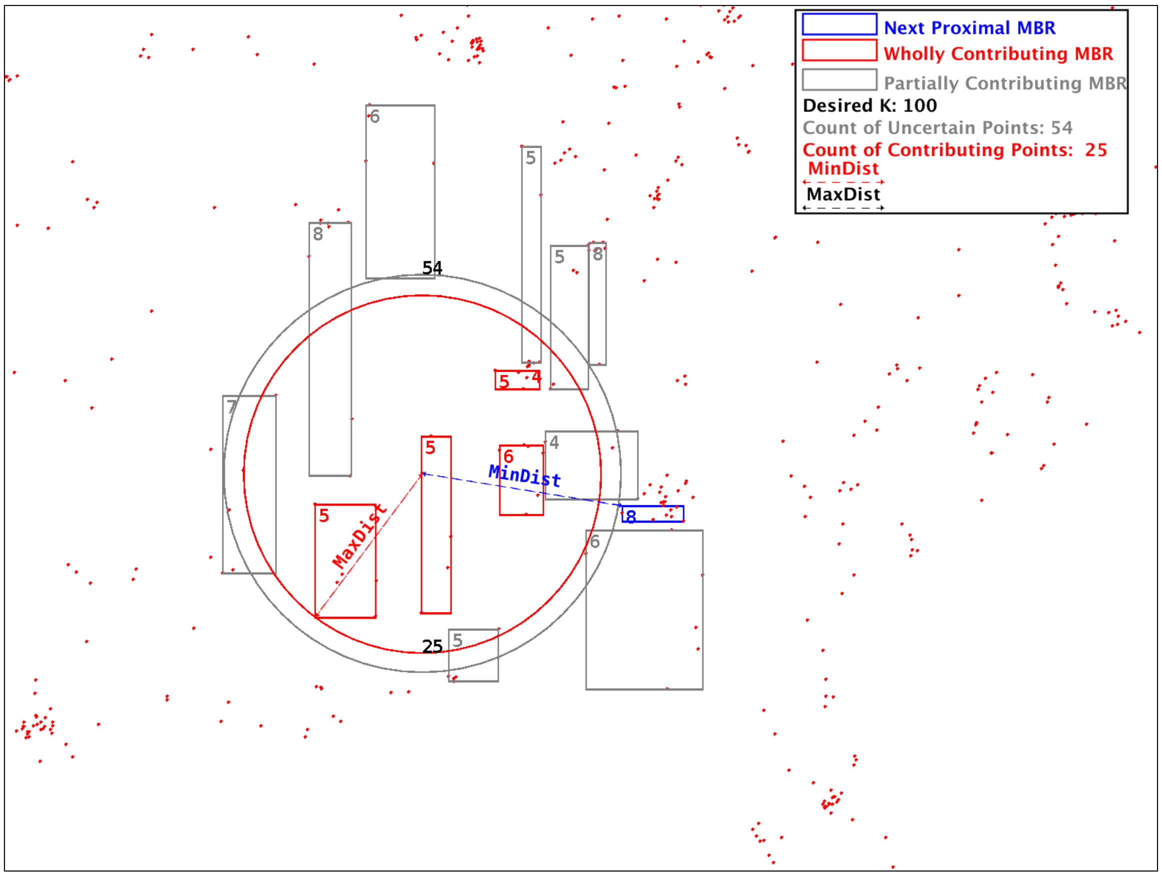


Figure A.14.: **Certain set** enlargement phase step 14: Since current total points (**certain** + **uncertain** = 79) is below (k = 100), **next proximal MBR**, based on MINDIST, will be added to the uncertain set.

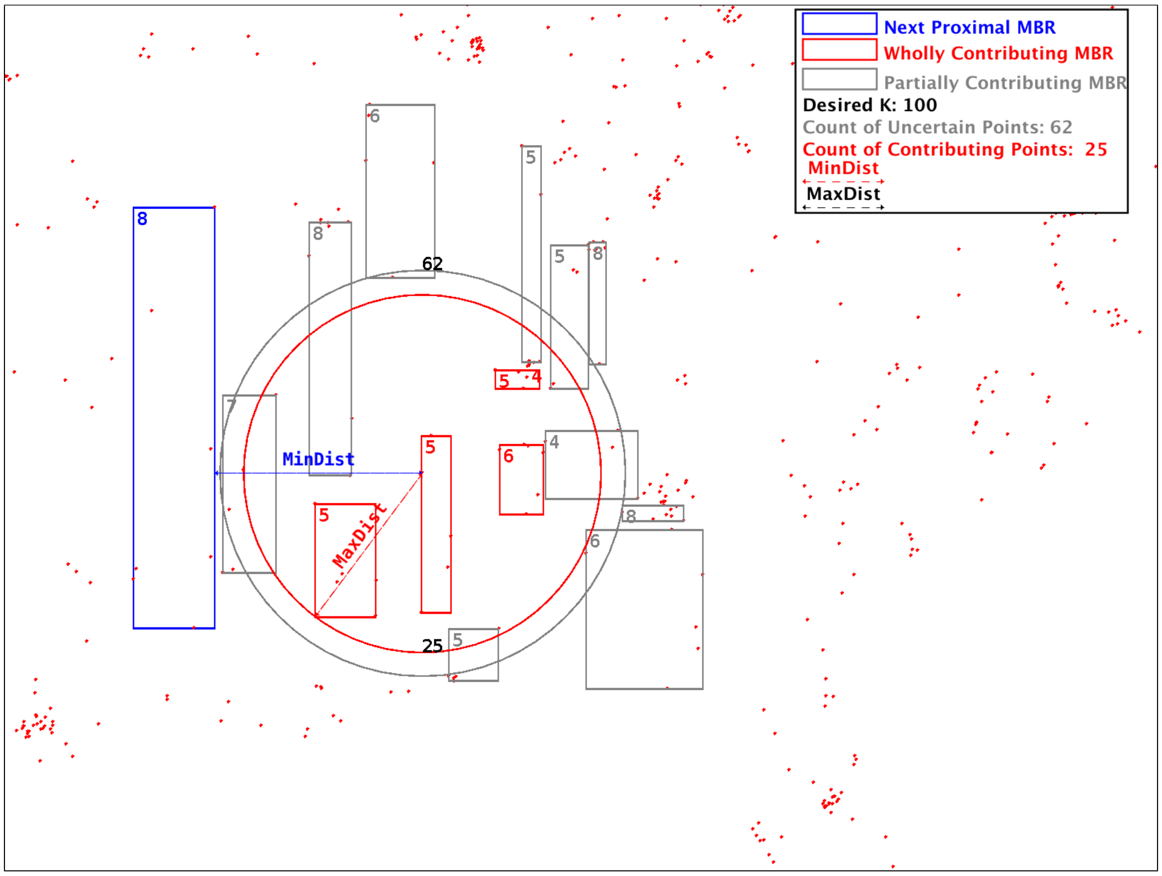

Next Proximal MBR
Wholly Contributing MBR
Partially Contributing MBR
Desired K: 100
Count of Uncertain Points: 62
Count of Contributing Points: 25
MinDist
MaxDist
62
25
MinDist
MaxDist

Figure A.15.: **Certain set** enlargement phase step 15: Since current total points (**certain** + **uncertain** = 87) is below (k = 100), **next proximal MBR**, based on MINDIST, will be added to the uncertain set.

.

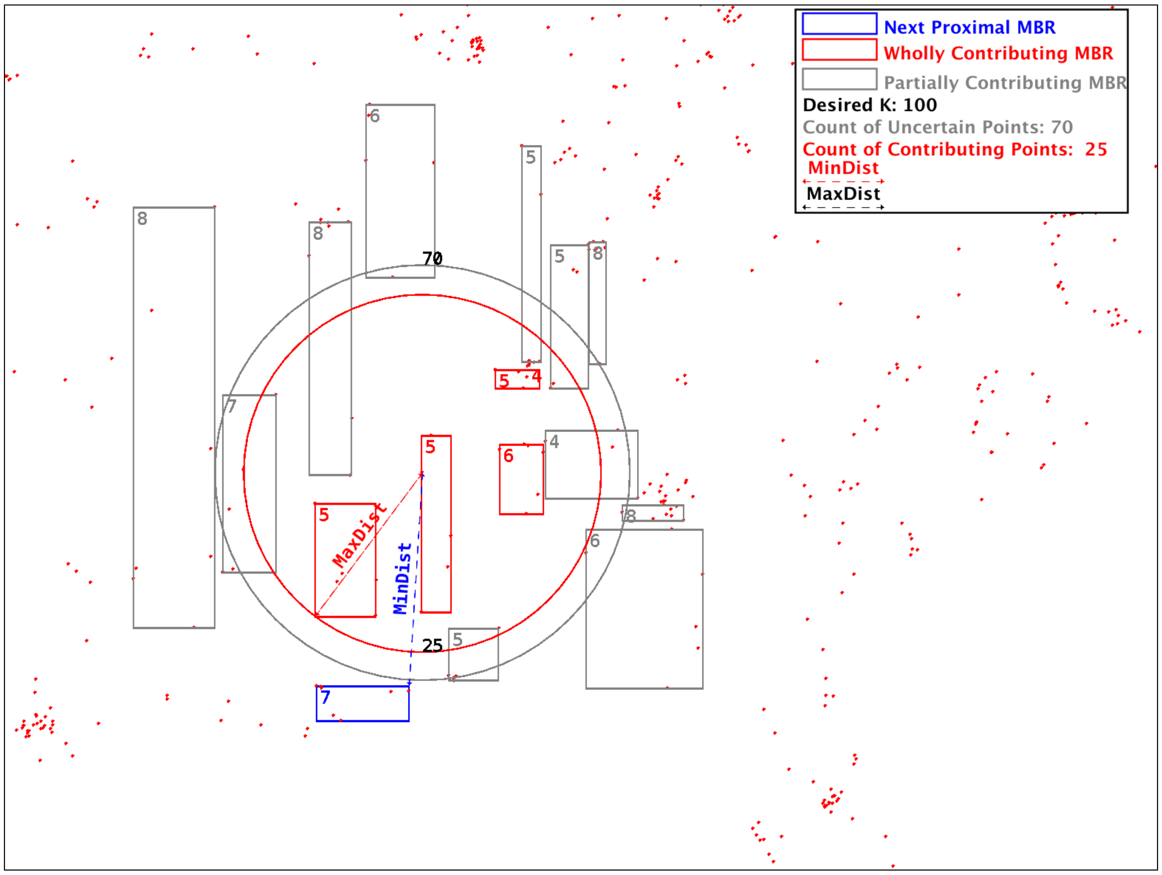


Figure A.16.: **Certain set** enlargement phase step 16: Since current total points (**certain** + **uncertain** = 95) is below (k = 100), **next proximal MBR**, based on MINDIST, will be added to the uncertain set.

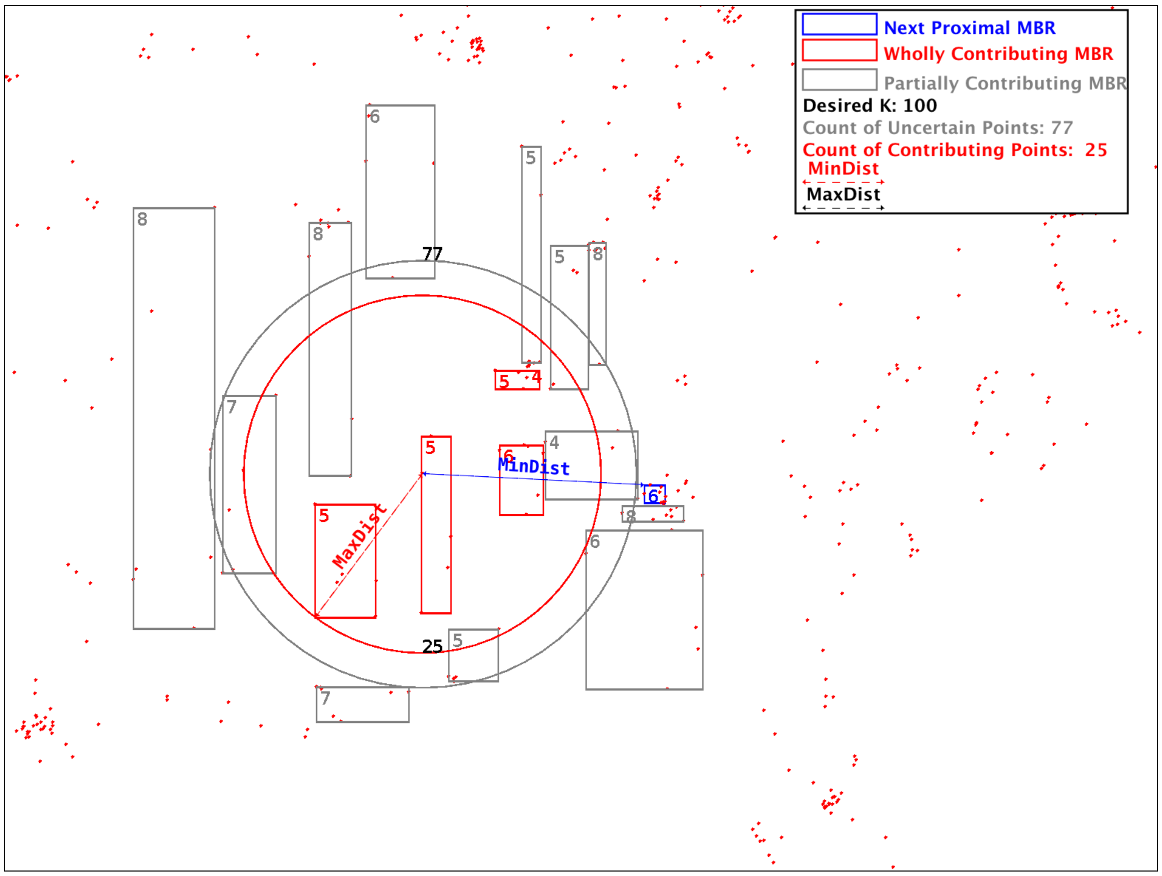

Next Proximal MBR
Wholly Contributing MBR
Partially Contributing MBR
Desired K: 100
Count of Uncertain Points: 77
Count of Contributing Points: 25
MinDist
MaxDist
77
25
MinDist
MaxDist

Figure A.17.: **Certain set** enlargement phase step 17: **Next proximal MBR** will not be added to the **uncertain set** as total points (**certain** + **uncertain** = 102) now exceeds ($k = 100$).

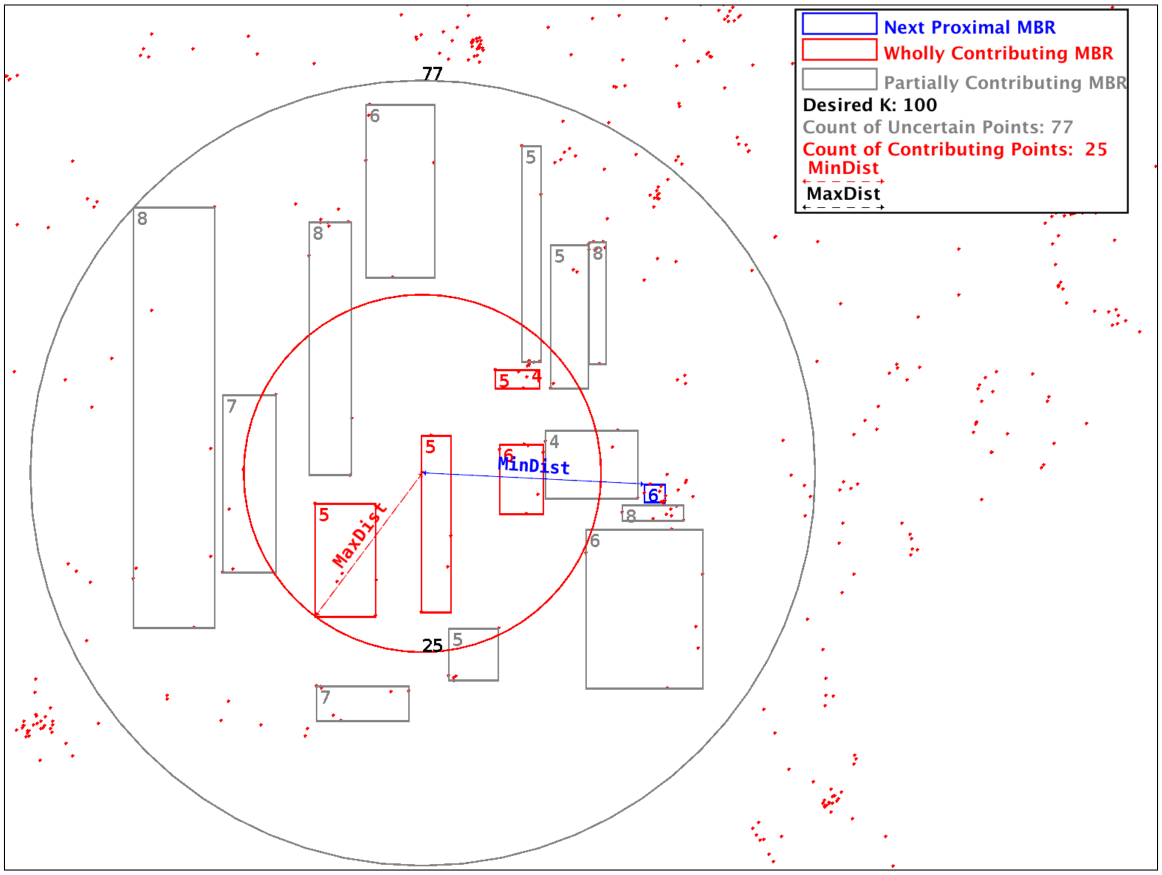

Next Proximal MBR
Wholly Contributing MBR
Partially Contributing MBR
Desired K: 100
Count of Uncertain Points: 77
Count of Contributing Points: 25
MinDist
MaxDist
77
25
MinDist
MaxDist

Figure A.18.: **Certain set** enlargement phase step 18: Total points (**certain** + **uncertain** = 102) now exceeds (k = 100). We now consider MAXDIST of the uncertain set (outer most circle).

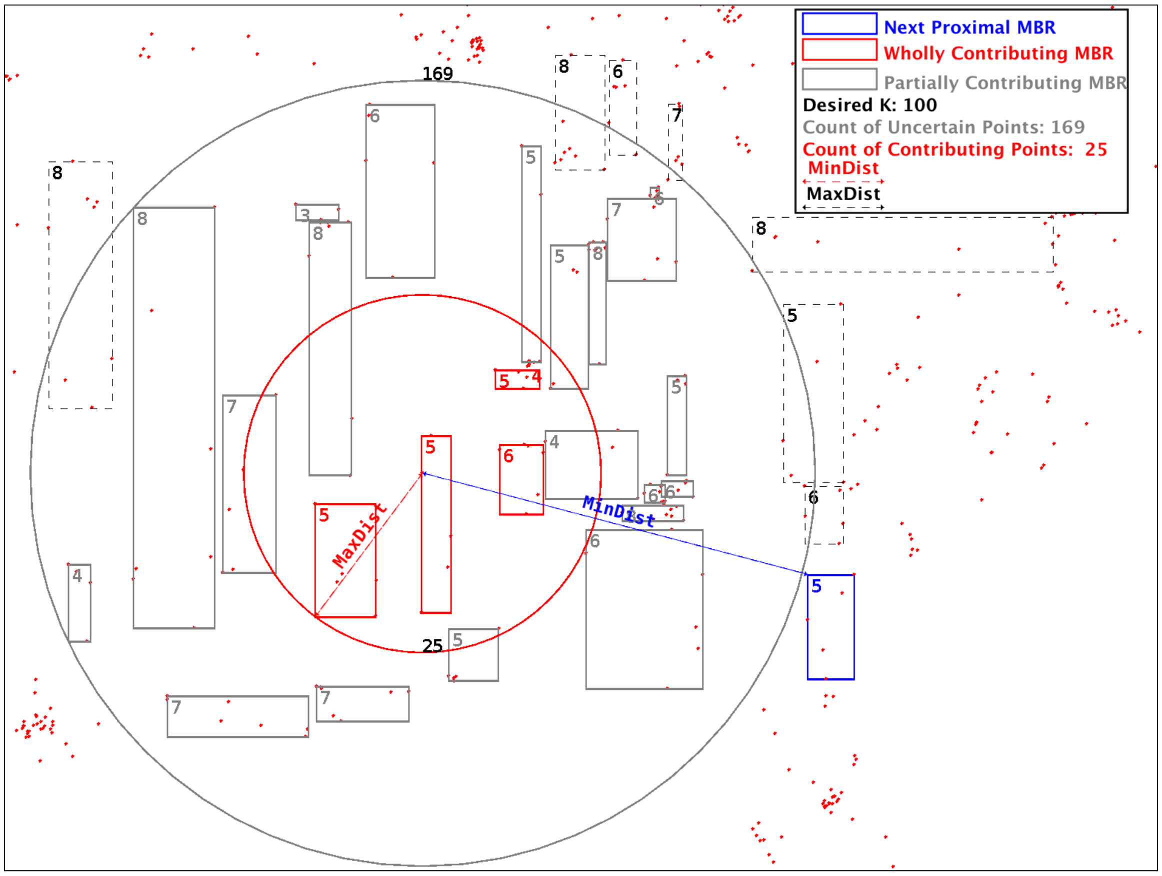

Next Proximal MBR
Wholly Contributing MBR
Partially Contributing MBR
Desired K: 100
Count of Uncertain Points: 169
Count of Contributing Points: 25
MinDist
MaxDist
169
25
MaxDist
MinDist

Figure A.19.: **Certain set** enlargement phase step 19

For correctness, we add all MBRs (dotted) overlapping with MAXDIST of the **uncertain set**. This expands the uncertain set to 169 points. The next phase minimizes the uncertain set using the MAXDIST metric unlike MINDIST used in current phase

A.2 Uncertain set minimization phase

In this phase, we start with the already obtained **certain set** from phase 1. We start with an empty **uncertain set** which we expand by distance browsing based on MAXDIST metric. Once (**certain** + **uncertain**) exceeds k we stop. For correctness, we add any MBR that is rooted inside the MAXDIST of the **uncertain set**. This is demonstrated graphically in figures A.20 to A.34.

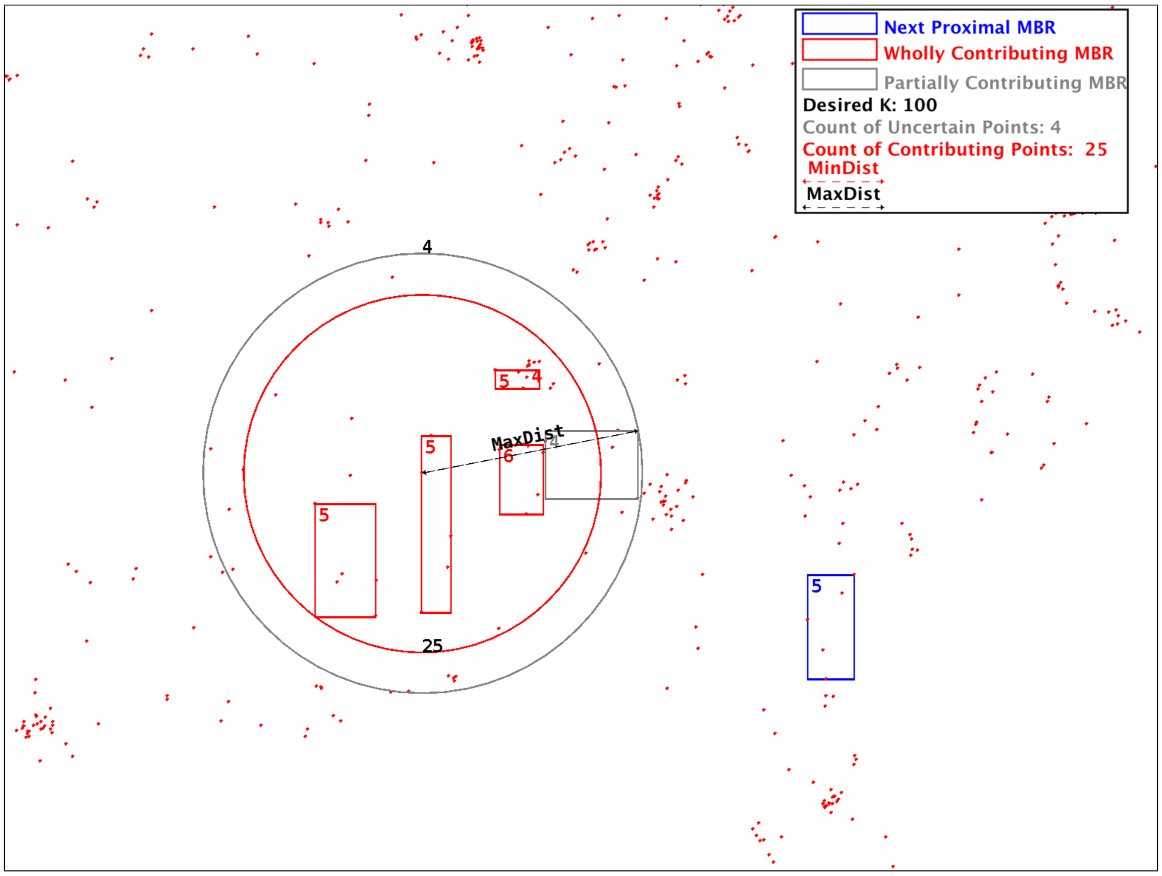

Next Proximal MBR
Wholly Contributing MBR
Partially Contributing MBR
Desired K: 100
Count of Uncertain Points: 4
Count of Contributing Points: 25
MinDist
MaxDist
4
25
MaxDist
5
4
5
6
4
5
5

Figure A.20.: **Uncertain set** minimization phase step 1: Since current total points (**certain** + **uncertain** = 25) is below (k = 100), We enlarge the **uncertain set** ,based on MAXDIST, for total points to be 29.

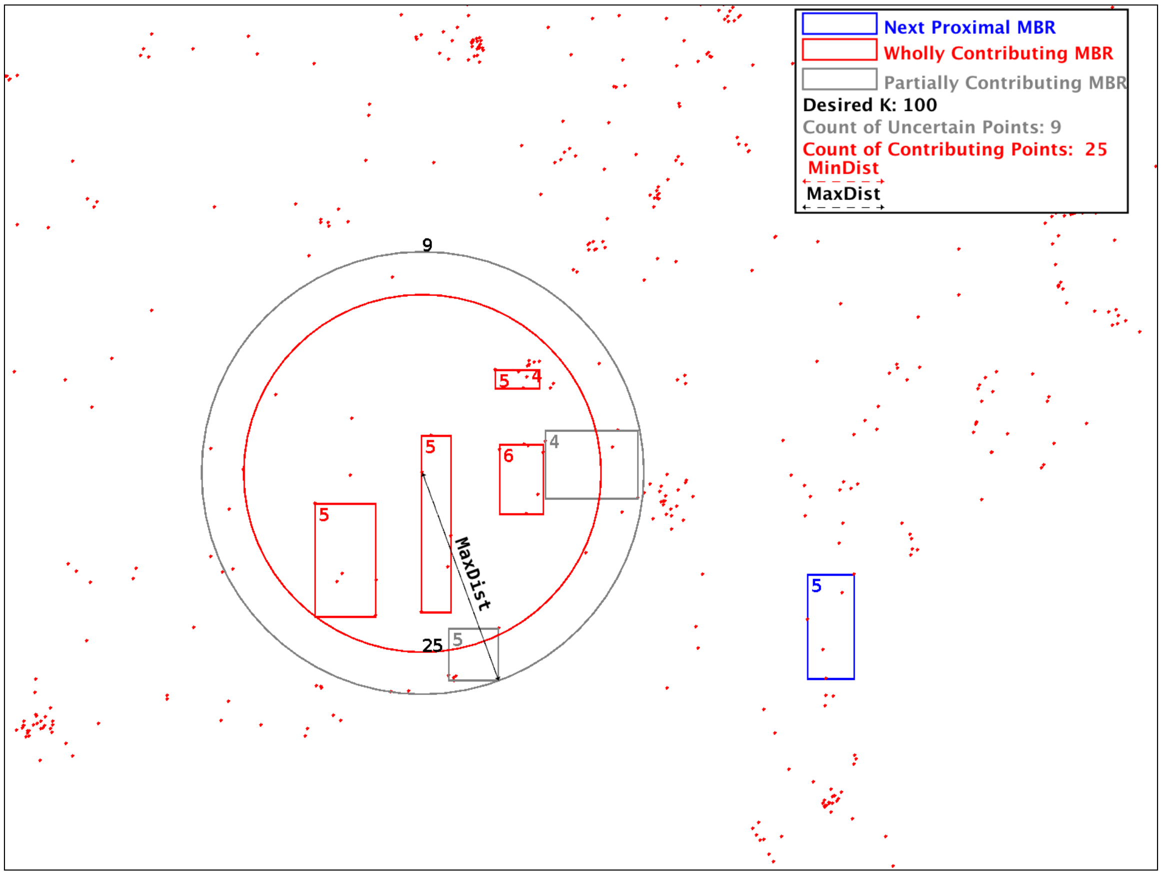

Next Proximal MBR
Wholly Contributing MBR
Partially Contributing MBR
Desired K: 100
Count of Uncertain Points: 9
Count of Contributing Points: 25
MinDist
MaxDist
9
25
MaxDist
5
4
5
6
4
5
5
5

Figure A.21.: **Uncertain set** minimization phase step 2: Since current total points (**certain** + **uncertain** = 29) is below (k = 100), We enlarge the **uncertain set** ,based on MAXDIST, for total points to be 34.

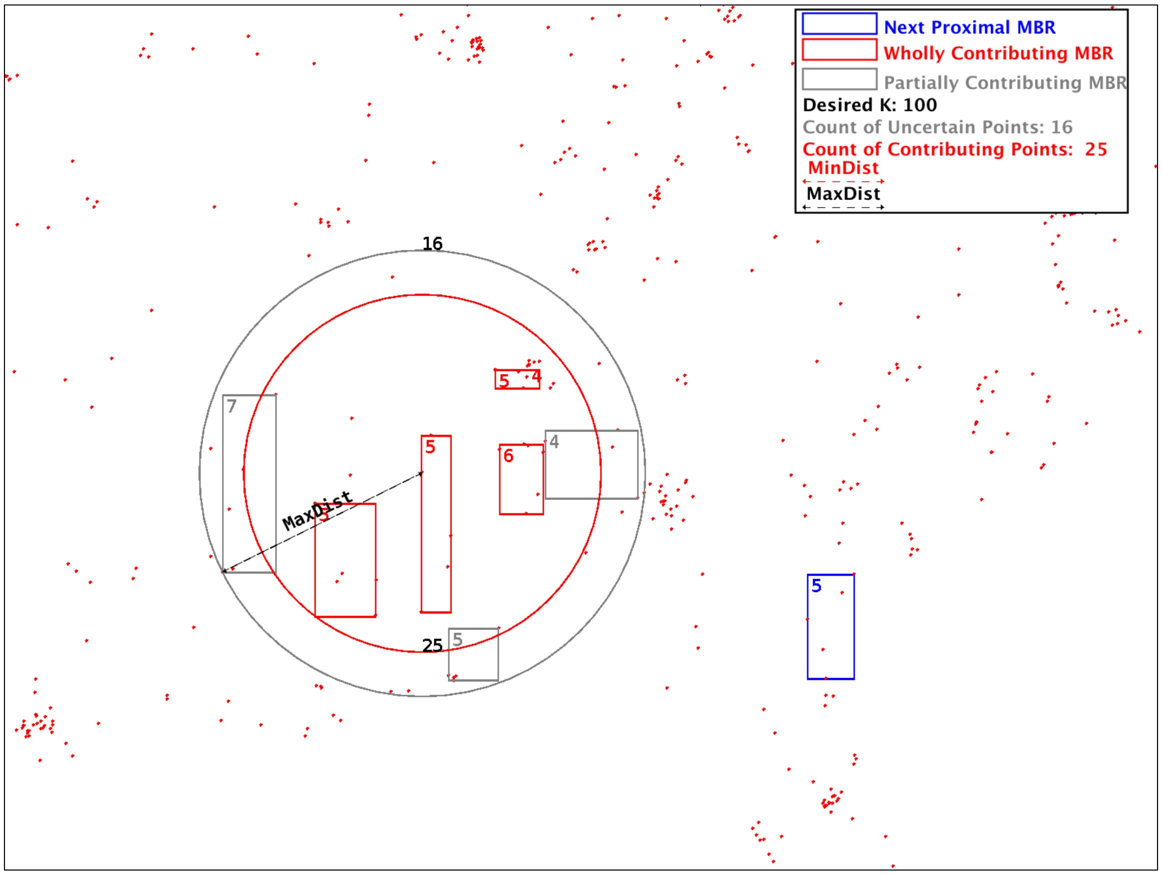

Next Proximal MBR
Wholly Contributing MBR
Partially Contributing MBR
Desired K: 100
Count of Uncertain Points: 16
Count of Contributing Points: 25
MinDist
MaxDist
16
25
MaxDist
7
5
4
5
6
4
5
5

Figure A.22.: **Uncertain set** minimization phase step 3: Since current total points (**certain** + **uncertain** = 34) is below (k = 100), We enlarge the **uncertain set** ,based on MAXDIST, for total points to be 41.

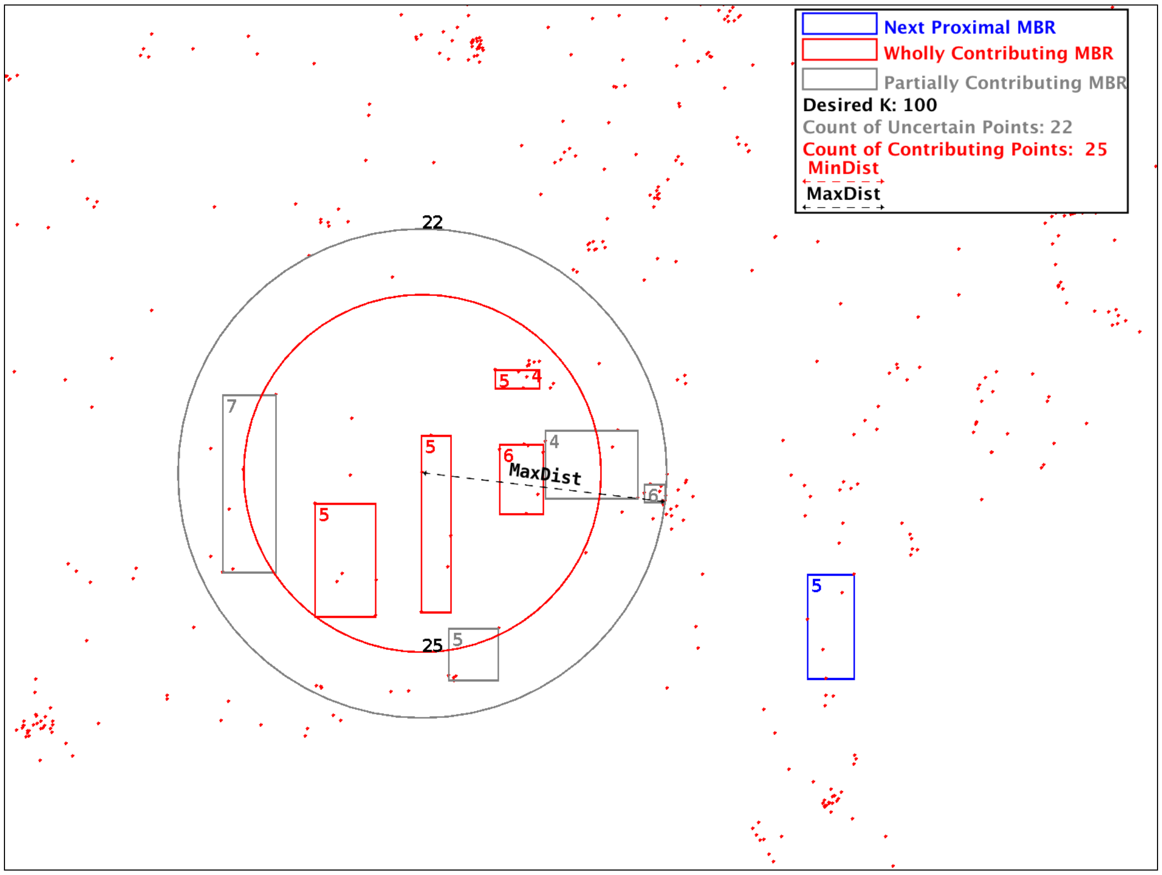


Figure A.23.: **Uncertain set** minimization phase step 4: Since current total points (**certain** + **uncertain** = 41) is below (k = 100), We enlarge the **uncertain set** ,based on MAXDIST, for total points to be 47.

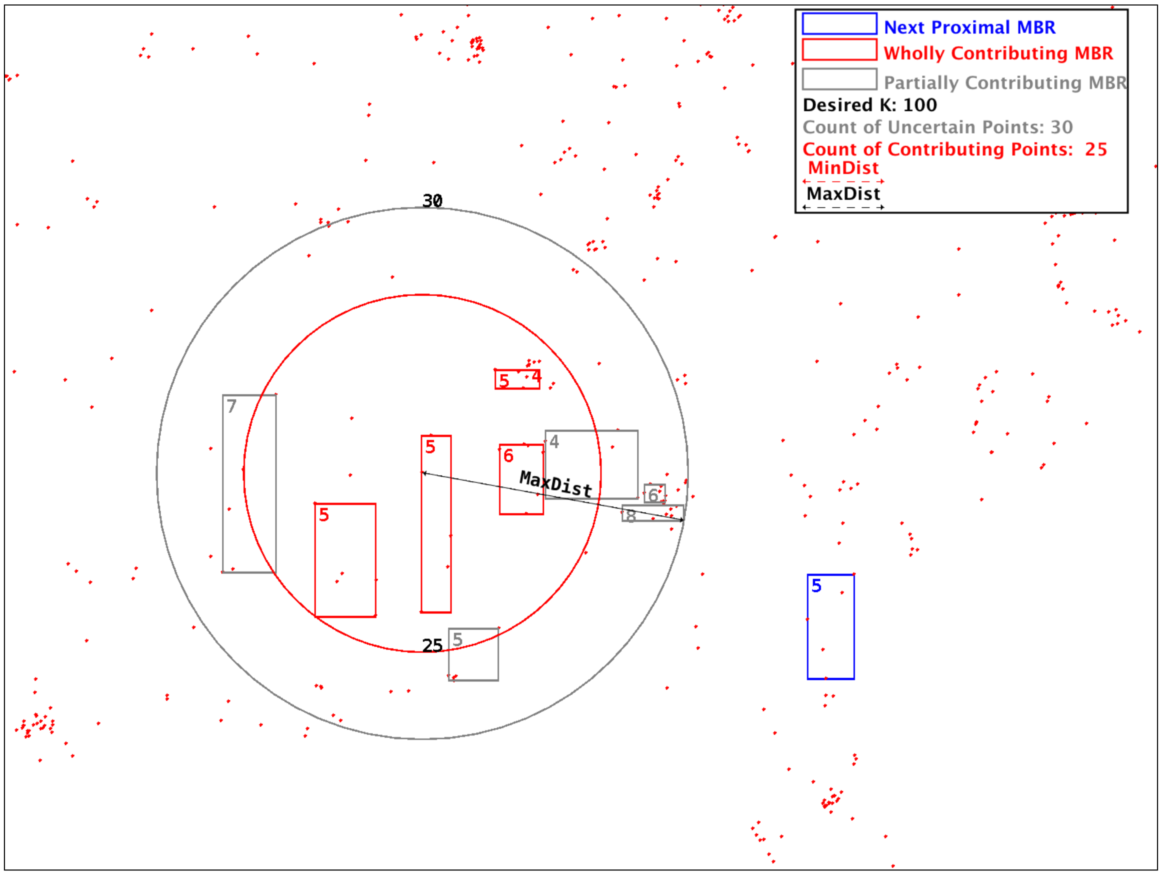


Figure A.24.: Uncertain set minimization phase step 5: Since current total points (**certain** + **uncertain** = 47) is below (k = 100), We enlarge the **uncertain set** ,based on MAXDIST, for total points to be 55.

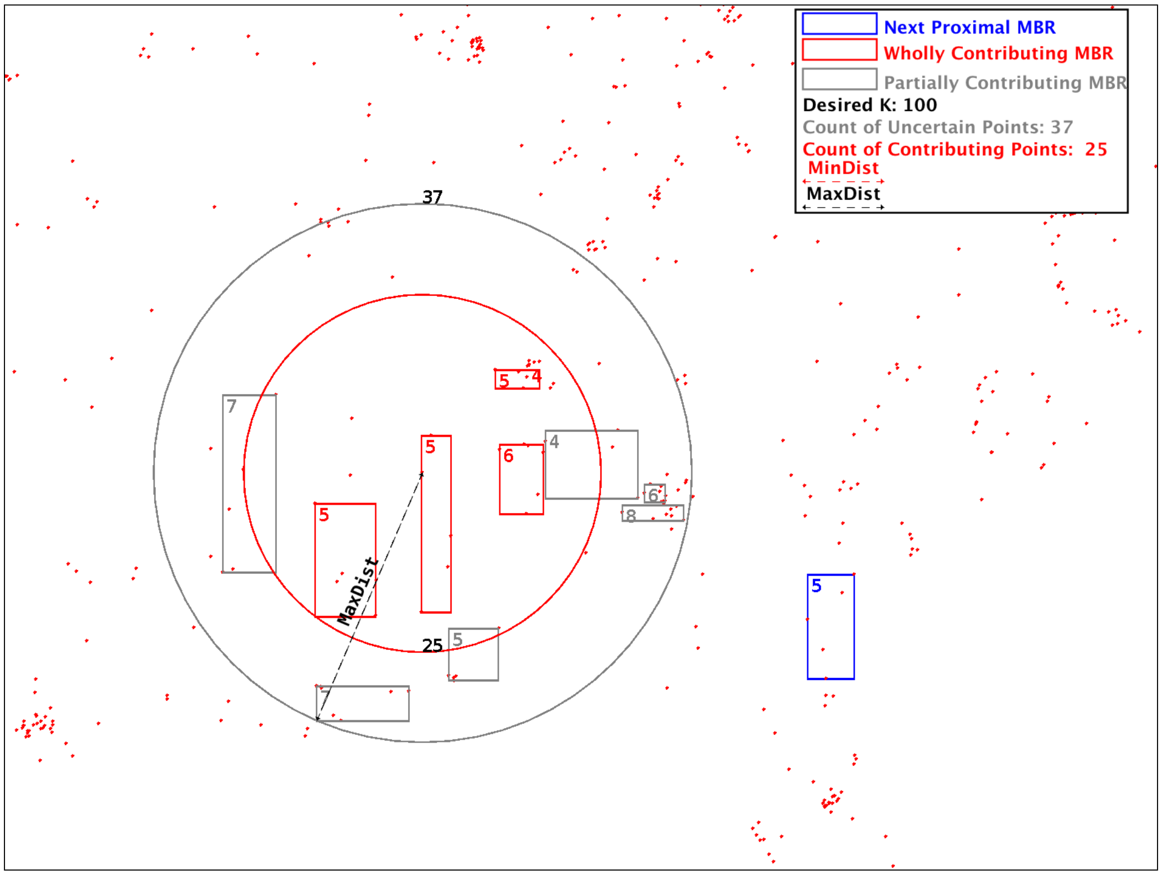


Figure A.25.: **Uncertain set** minimization phase step 6: Since current total points (**certain** + **uncertain** = 55) is below (k = 100), We enlarge the **uncertain set** ,based on MAXDIST, for total points to be 62.

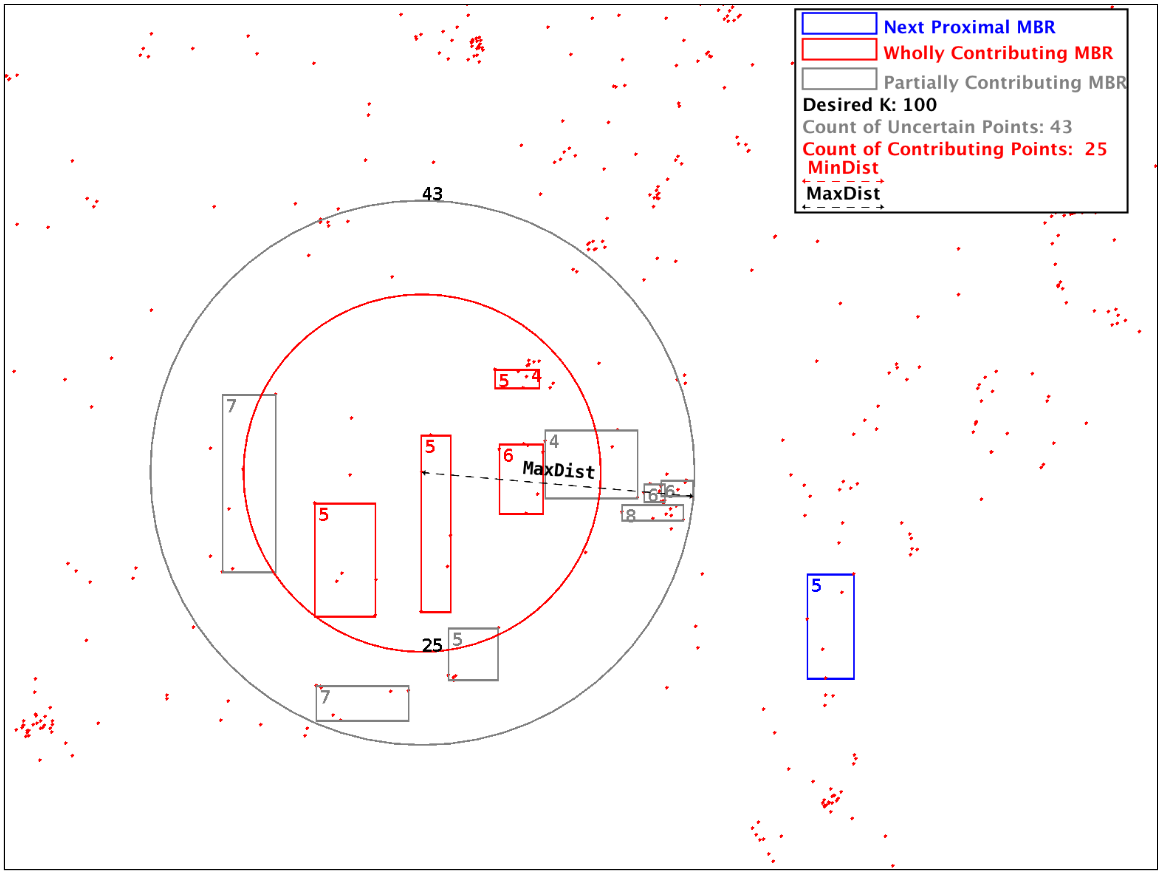


Figure A.26.: Uncertain set minimization phase step 7: Since current total points (**certain** + **uncertain** = 62) is below (k = 100), We enlarge the **uncertain set** ,based on MAXDIST, for total points to be 68.

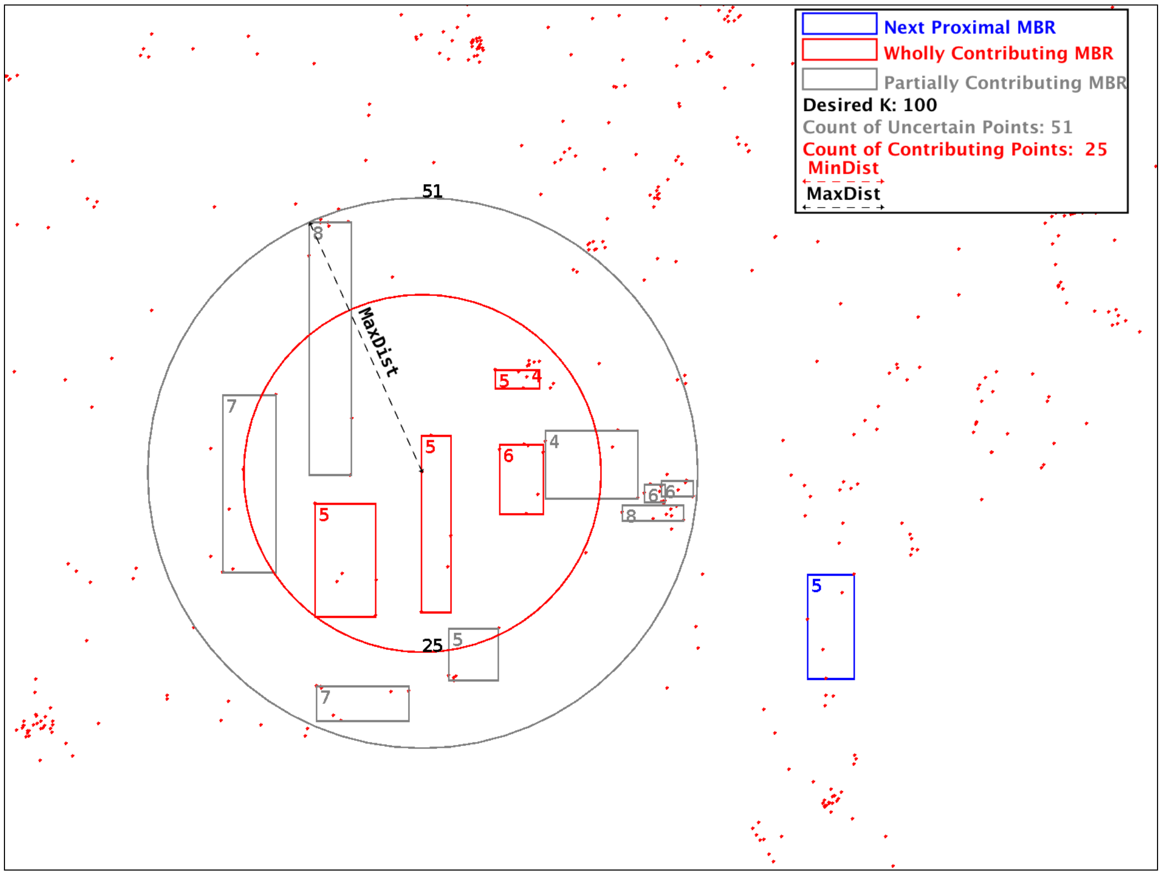


Figure A.27.: **Uncertain set** minimization phase step 8: Since current total points (**certain** + **uncertain** = 68) is below (k = 100), We enlarge the **uncertain set** ,based on MAXDIST, for total points to be 76.

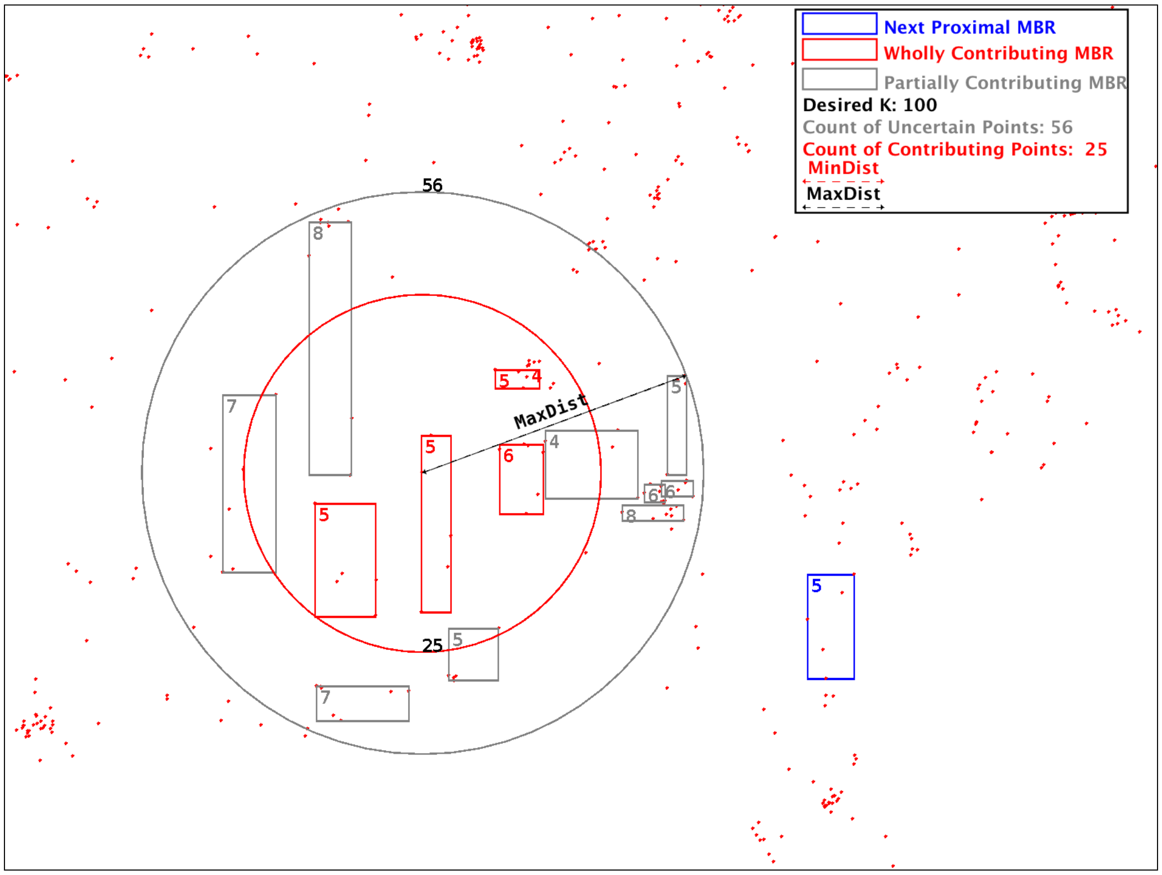


Figure A.28.: **Uncertain set** minimization phase step 9: Since current total points (**certain** + **uncertain** = 76) is below (k = 100), We enlarge the **uncertain set** ,based on MAXDIST, for total points to be 81.

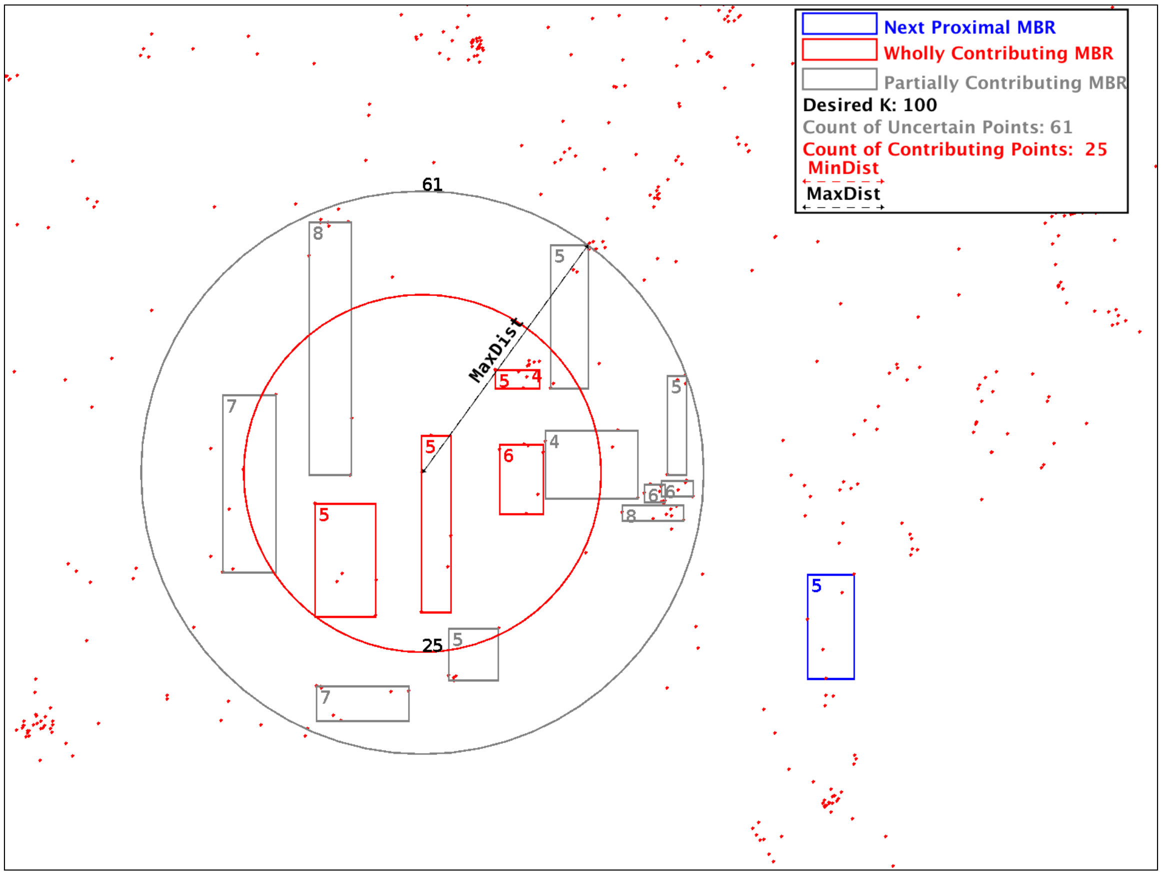


Figure A.29.: **Uncertain set** minimization phase step 10: Since current total points (**certain** + **uncertain** = 81) is below (k = 100), We enlarge the **uncertain set** ,based on MAXDIST, for total points to be 86.

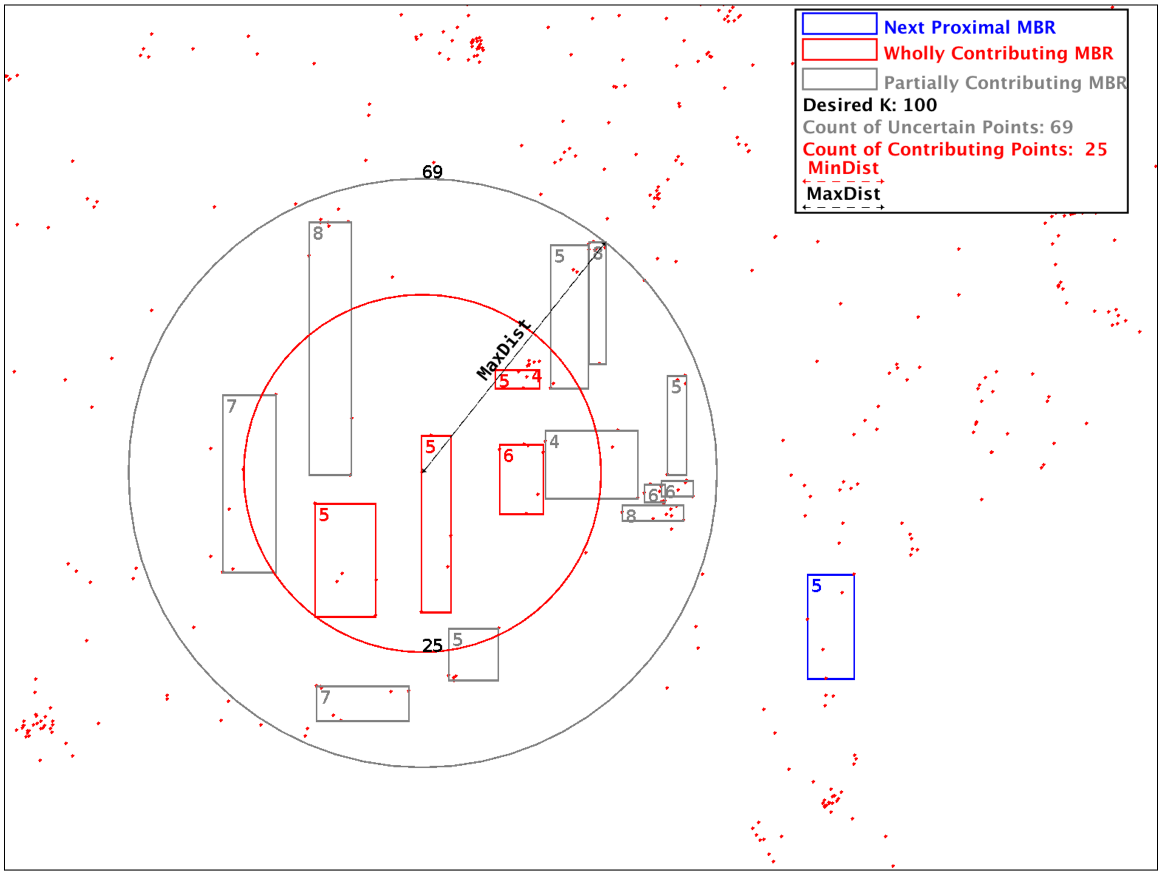


Figure A.30.: **Uncertain set** minimization phase step 11: Since current total points (**certain** + **uncertain** = 86) is below (k = 100), We enlarge the **uncertain set** ,based on MAXDIST, for total points to be 94.

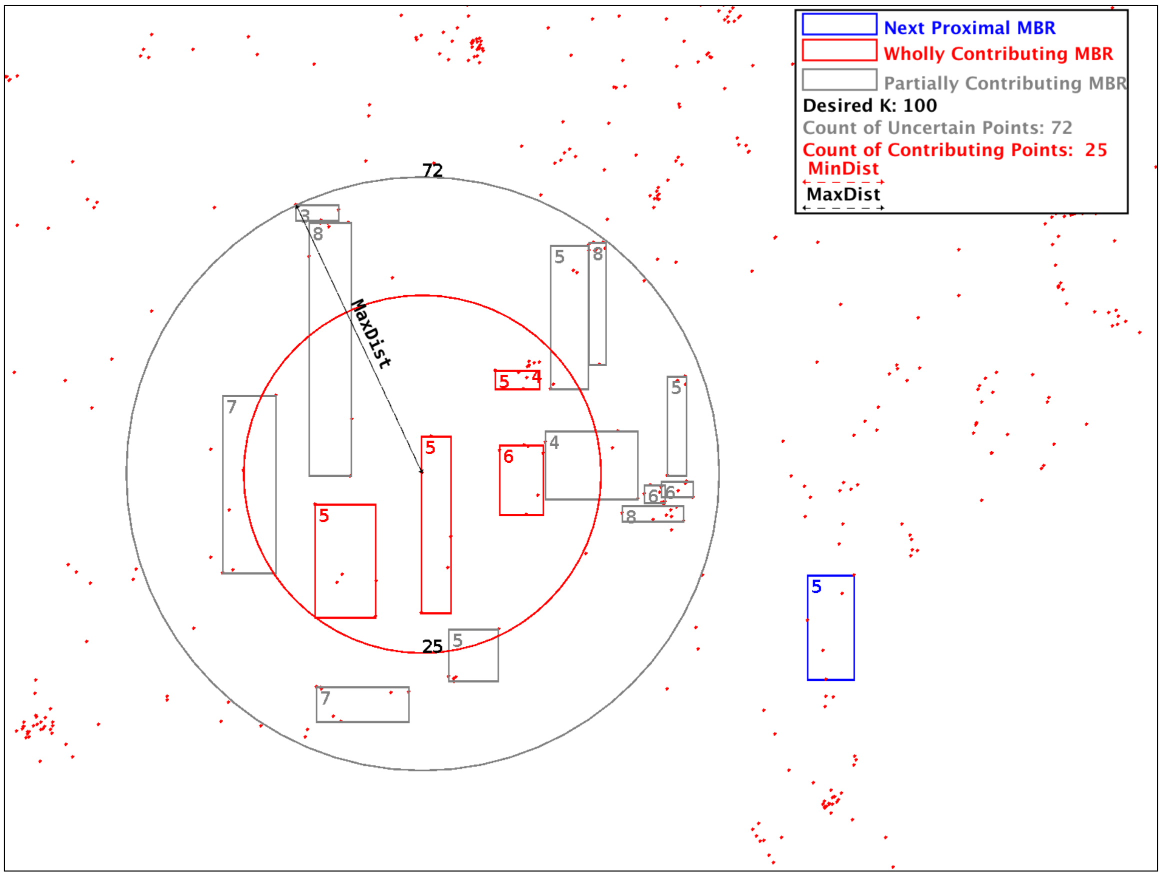

Next Proximal MBR
Wholly Contributing MBR
Partially Contributing MBR
Desired K: 100
Count of Uncertain Points: 72
Count of Contributing Points: 25
MinDist
MaxDist
72
25
MaxDist

Figure A.31.: Uncertain set minimization phase step 12: Since current total points (**certain** + **uncertain** = 94) is below (k = 100), We enlarge the **uncertain set** ,based on MAXDIST, for total points to be 97.

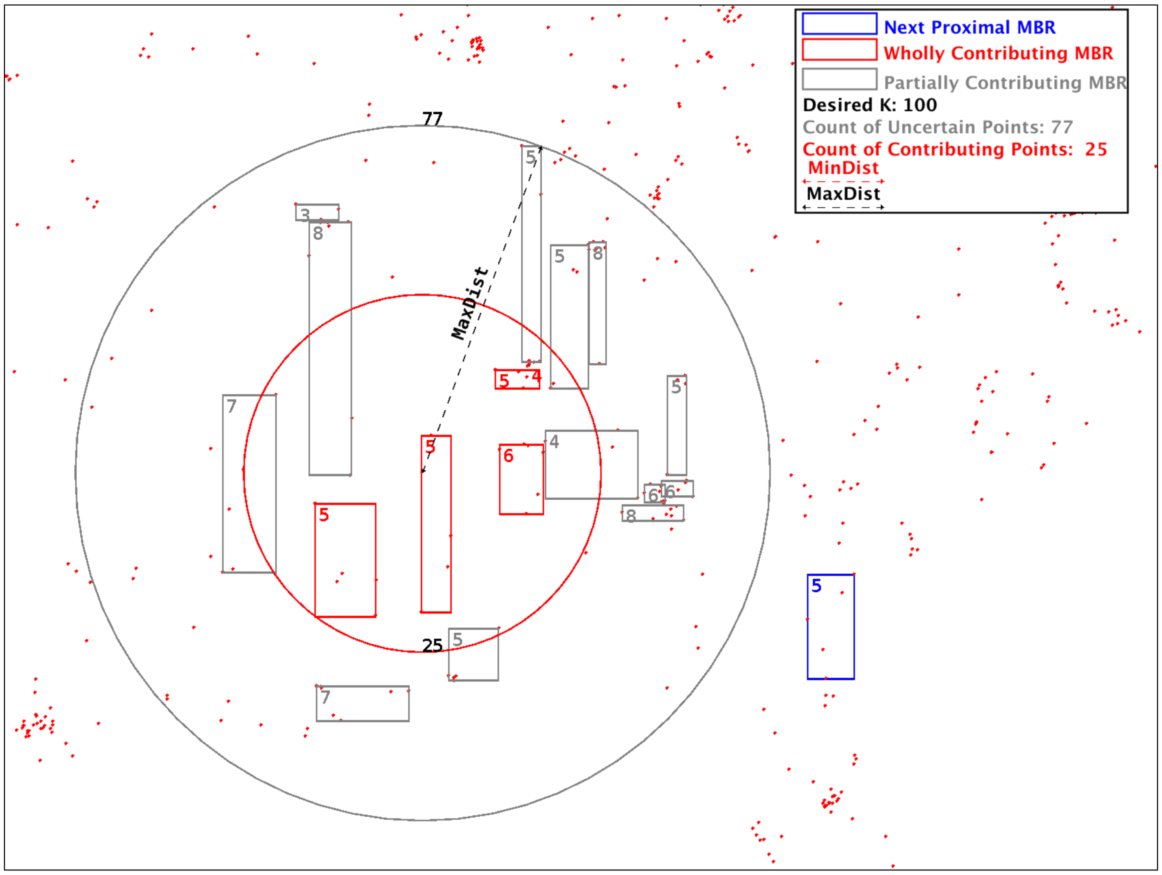

Next Proximal MBR
Wholly Contributing MBR
Partially Contributing MBR
Desired K: 100
Count of Uncertain Points: 77
Count of Contributing Points: 25
MinDist
MaxDist
77
25
MaxDist

Figure A.32.: **Uncertain set** minimization phase step 13: Since current total points (**certain** + **uncertain** = 97) is below (k = 100), We enlarge the **uncertain set** ,based on MAXDIST, for total points to be 102.. Thus exceeding k.

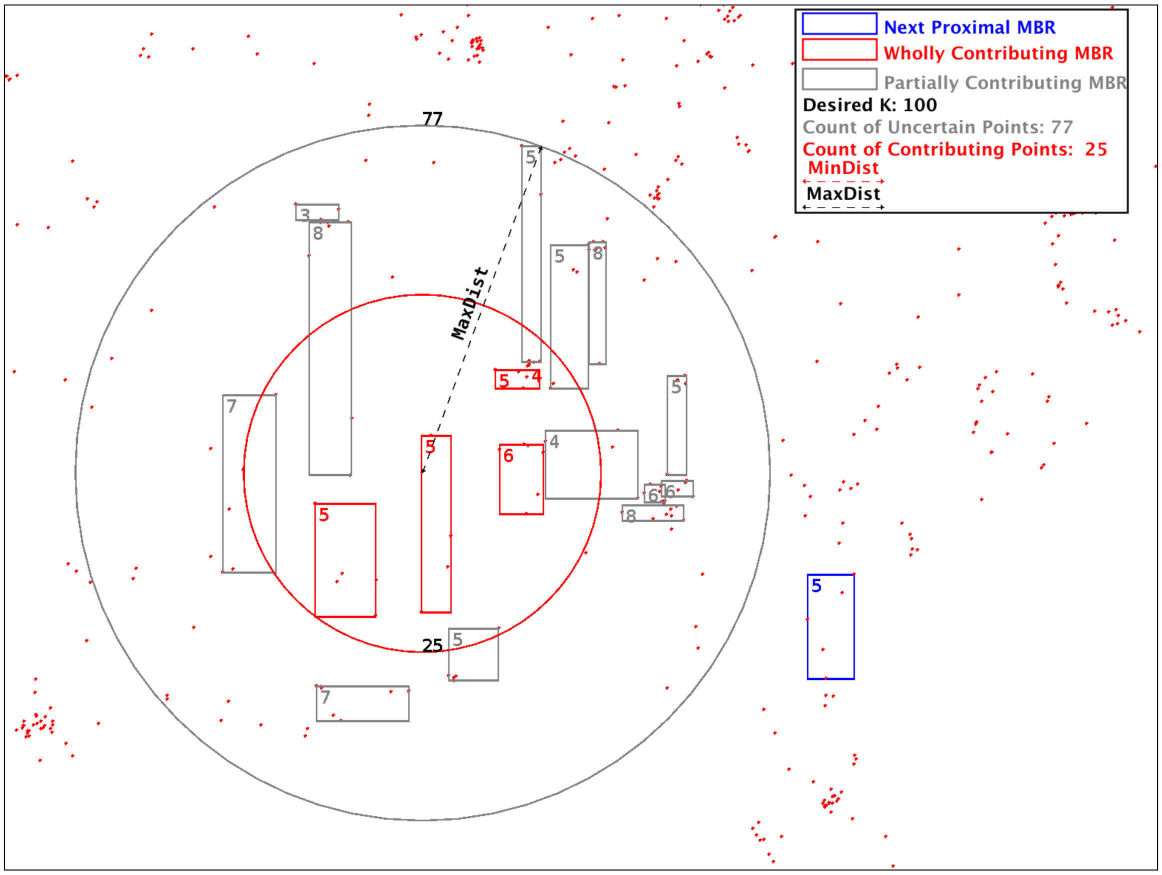

Next Proximal MBR
Wholly Contributing MBR
Partially Contributing MBR
Desired K: 100
Count of Uncertain Points: 77
Count of Contributing Points: 25
MinDist
MaxDist
77
25
MaxDist

Figure A.33.: **Uncertain set** minimization phase step 14: Since current total points (**certain**+**uncertain** = 102) exceeds ($k = 100$)), we stop expansion. For correctness, see next step.

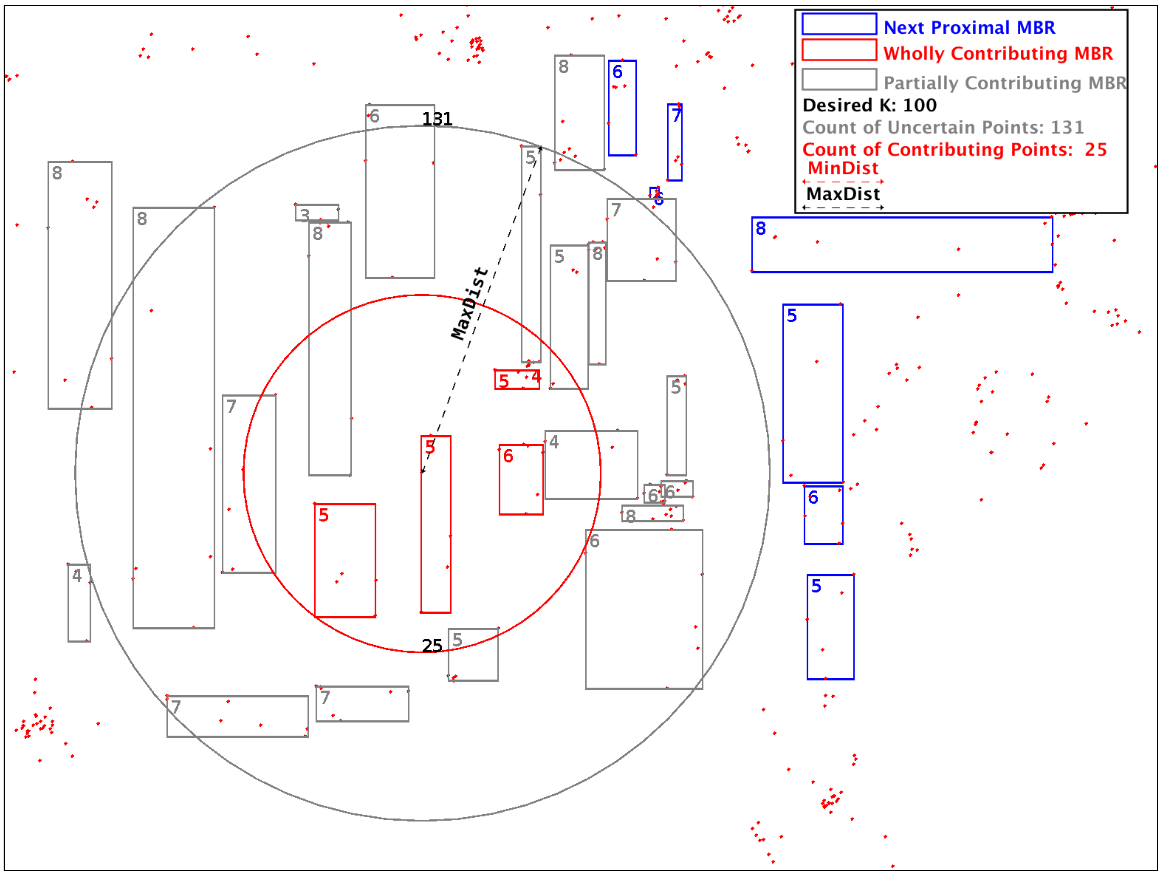

Next Proximal MBR
Wholly Contributing MBR
Partially Contributing MBR
Desired K: 100
Count of Uncertain Points: 131
Count of Contributing Points: 25
MinDist
MaxDist
131
25
MaxDist

Figure A.34.: **Uncertain set** minimization phase step 15: Once we reach (**certain** + **uncertain** $> k$, and for correctness, we add all MBRs that are rooted inside the uncertain MAXDIST circle of the **uncertain set**. Thus the uncertain count is expanded from 77 to 131. 131 is less than what we had at end of phase 1 (169). The saving (pruned MBRs) are highlighted in blue.