A technique for adding range restrictions to generalized searching problems

Prosenjit Gupta* Ravi Janardan† Michiel Smid‡

Abstract

In a generalized searching problem, a set S of n colored geometric objects has to be stored in a data structure, such that for any given query object q, the distinct colors of the objects of S intersected by q can be reported efficiently. In this paper, a general technique is presented for adding a range restriction to such a problem. The technique is applied to the problem of querying a set of colored points (resp. fat triangles) with a fat triangle (resp. point). For both problems, a data structure is obtained having size $O(n^{1+\epsilon})$ and query time $O((\log n)^2 + C)$. Here, C denotes the number of colors reported by the query, and ϵ is an arbitrarily small positive constant.

Keywords: Computational geometry, data structures, intersection searching, range restriction.

1 Introduction

Geometric searching problems arise in a large variety of application areas, such as computer graphics, robotics, VLSI layout design, and databases. In such a problem, a set S of n geometric objects has to be stored in a data

*Bell Laboratories, 600 Mountain Avenue, Murray Hill NJ 07974, U.S.A. E-mail: prosenjit@lucent.com . Part of this work was done while at MPI-Informatik, Saarbrücken, Germany.

†Department of Computer Science, University of Minnesota, Minneapolis, MN 55455, U.S.A. E-mail: janardan@cs.umn.edu. This author was supported by a Grant–in–Aid of Research from the Graduate School of the University of Minnesota.

‡Department of Computer Science, University of Magdeburg, D-39106 Magdeburg, Germany. E-mail: michiel@isg.cs.uni-magdeburg.de . Part of this work was done while at MPI-Informatik, Saarbrücken, Germany, and the Department of Computer Science, King's College London, UK.

structure, such that for any given query object q, we can efficiently report all objects of S that intersect q. The efficiency of such a data structure is typically expressed by its size and query time, where the latter has the form $O(f(n) + K \cdot g(n))$, for some "small" functions f and g. In this expression, K denotes the number of elements of S that intersect the query object. We call these problems *standard searching problems*, in order to distinguish them from *generalized searching problems*, which are the subject of this paper.

In many applications, a more general form of searching problem arises: The objects in S come aggregated in disjoint groups and of interest are questions regarding the intersection of q with the groups rather than with the objects. Here, we say that q intersects a group if and only if it intersects at least one object in the group. We will associate with each group a different color and imagine that all the objects in the group have that color. Then, in the *generalized searching problem*, we want to report the distinct colors of those elements of S that intersect q. Note that the generalized problem reduces to the standard one when each color class has cardinality one. For applications of these generalized searching problems, see [1, 4, 6, 8].

We can solve a generalized searching problem by first determining the objects of S that intersect q and then reading off the distinct colors. However, the query time can be very high since q could intersect $\Omega(n)$ objects but only $O(1)$ distinct colors. For a generalized searching problem, we want to obtain query times that are sensitive to the number C of distinct colors intersected, rather than K. Typically, our goal is to obtain query times of the form $O(f(n)+C)$ or $O(f(n)+C \cdot g(n))$, where f and g are polylogarithmic.

Generalized searching problems first appeared in [6]. Subsequently, several papers were published on this type of problem. See [1, 3, 4, 5, 7, 8].

In this paper, we consider the problem of transforming generalized searching problems into other problems by *adding a range restriction*. This transformation was introduced for *standard* searching problems by Bentley [2].

Let PR denote a generalized searching problem on a set S of objects. To add a range restriction to PR, we give each object p in S an additional parameter $k_p \in \mathbb{R}$. In the transformed searching problem, we only query objects in S that have their parameter in a given range. We define this more precisely.

Definition 1 *Consider a generalized searching problem PR on a set S of objects with a query object q drawn from a class Q of query objects of some type. We denote this as $PR(q, S)$ with the understanding that q is not fixed but ranges over all objects of Q. To add a* range restriction, *we associate*

with each object p in S a real number k_p. In the transformed generalized searching problem TPR, a query consists of a query object q together with an interval $[a, b]$, and

$$TPR(q, [a, b], S) := PR(q, \{p \in S : a \leq k_p \leq b\}).$$

In the case where the range restriction is of the form $(-\infty, b]$ or $[a, \infty)$, we speak about a half-infinite range restriction.

As an example, consider the d-dimensional *generalized orthogonal range searching problem*, in which we are given a set S of n colored points in $\mathbb{R}^d$. The query is an axes-parallel d-box q, and we want to report the distinct colors of those points that are contained in q. This problem is obtained by adding a range restriction to the $(d-1)$-dimensional generalized range searching problem.

We remark that generalized searching problems are not *decomposable* in the sense of Bentley [2]: Because duplicate colors must be eliminated, the solution to the original problem cannot be obtained in constant time from the solutions to subproblems.

As mentioned, the notion of adding a range restriction (for standard problems) first appeared in Bentley [2]. He gave a general technique that transforms a data structure for solving the standard problem PR into a data structure for the standard problem TPR. Later, other general techniques were developed by Willard and Lueker [11], Scholten and Overmars [10], van Kreveld [8], and Lenhof and Smid [9]. With the exception of [8], these techniques can be applied to generalized searching problems, but this leads to data structures in which each color may be reported a logarithmic number of times. (That is, using the notation above, we have $g(n) = \Theta(\log n)$.) The results in [8] provide structures where each color is reported at most a constant number of times. We discuss this in more detail below.

The main result of this paper is a general technique that transforms data structures DS and TDS solving the generalized searching problems PR and TPR, respectively, into a data structure TDS' solving TPR that uses much less space than TDS does. More precisely, if we start with (i) a data structure DS having $O(\text{polylog}(n) + C)$ query time and using $O(n^{1+\epsilon})$ space, and (ii) a data structure TDS having $O(\text{polylog}(n) + C)$ query time and using (possibly high) polynomial space, and apply the transformation several times, we get a data structure for TPR having $O(\text{polylog}(n) + C)$ query time and using only $O(n^{1+\epsilon})$ space. Here, ϵ is an arbitrarily small positive constant.

Generalized Problem	*Space*	*Query Time*	*Reference*
Querying Points with Fat Triangles	$n(\log n)^3$	$(\log n)^4 + C(\log n)^2$	[5]
	$n^{1+\epsilon}$	$(\log n)^2 + C$	this paper
Querying Fat Triangles with Points	$n(\log n)^2$	$(\log n)^3 + C \log n$	[5]
	$n^{1+\epsilon}$	$(\log n)^2 + C$	[5], this paper

Table 1: *Overview of results for generalized problems on fat triangles. All bounds given are "big–oh". C denotes the output size, and ϵ is an arbitrarily small positive constant.*

We remark that in earlier work, van Kreveld presented an alternative approach for obtaining $O(n^{1+\epsilon})$ space and $O(\text{polylog}(n) + C)$ query time for range-restricted colored problems; see Theorem 5.1(ii), Theorem 5.3(ii), and Corollary 7.3(i) in [8]. His work is based on multiway balanced search trees with secondary structures. Our approach is different: we construct TDS' iteratively from DS and TDS by using a "bootstrapping" approach, which is interesting in its own right.

We illustrate our general technique by giving efficient data structures for the following two problems: Store a set of colored points (resp. fat triangles) in a data structure, such that for any query fat triangle (resp. point) q, we can report the distinct colors of those points (resp. triangles) that are contained in (resp. contain) q. Here, a triangle is called *fat*, if all its angles are at least equal to some fixed constant α. To our knowledge, this problem was first considered in [5]. See Table 1 for this and our results in this paper. While some of our results were known already, the techniques in this paper allow us to now derive these results in a unified way.

The rest of this paper is organized as follows. In Section 2, we give the general technique of adding a range restriction to a generalized searching problem. In Section 3, we show how to apply this technique to generalized problems for fat triangles. We conclude the paper in Section 4 with some final remarks.

We now discuss one important issue that arises in the encoding and handling of colors. Throughout the paper, we assume that our algorithms incorporate the mechanisms that we describe here and so we will not repeat them afterwards. The number of colors for a given problem can range from 1 to n. We encode each color as an integer in the range $[1, n]$. This allows us to

use colors as array indices. In many of our generalized searching problems, when answering a query we may encounter the same color more than once (but no more than $O(1)$ times). Our goal is to eliminate the duplicate colors efficiently before we output the answer. We can do this by using an array, $A[1:n]$, of colors to keep track of the distinct colors that are found during a query. We also store the distinct colors found in a linked list. After the query, A can be reset in time proportional to the output size by scanning the list.

2 Adding a range restriction

Let S be a set of n colored objects, and let $PR(q, S)$ be a generalized searching problem for S with query object q. Let each object p of S have an additional parameter $k_p \in \mathbb{R}$. Let TPR be the generalized searching problem that is obtained by adding a range restriction to PR.

Assume we have a data structure DS that stores the set S, such that generalized queries $PR(q, S)$ can be solved in $O((\log n)^u + C)$ time, for some positive constant u. Let the size of DS be bounded by $O(n^{1+\epsilon})$, where ϵ is an arbitrarily small positive constant.

Also, assume we have a data structure TDS for the set S, such that generalized queries $TPR(q, [a : \infty), S)$ can be solved in $O((\log n)^v + C)$ time, for some positive constant v. Let the size of TDS be bounded by $O(n^w)$ for some constant $w > 1$.

We will first show how to construct a data structure that solves generalized queries $TPR(q, [a : \infty), S)$ in $O((\log n)^{\max(u,v,1)} + C)$ time, using $O((n^{1+\epsilon})$ space, for an arbitrarily small positive constant ϵ. Then, we will show how to solve generalized queries $TPR(q, [a : b], S)$ within the same time and space bounds.

The basic transformation: We sort and store the elements of S in non-increasing order of their parameter k_p. Let the sorted set be $S = \{p_1, p_2, \ldots, p_n\}$, i.e., $k_{p_1} \geq k_{p_2} \geq \ldots \geq k_{p_n}$.

Let m be a parameter such that $1 \leq m \leq n$. For $0 \leq i < n/m$, let

$$S_i = \{p_1, p_2, \ldots, p_{im}\},$$

and

$$S'_i = \{p_{im+1}, p_{im+2}, \ldots p_{(i+1)m}\}.$$

For each i, $0 \leq i < n/m$, we store the set S_i in a data structure DS_i (of type DS) for solving generalized queries of the form $PR(q, S_i)$. Moreover, we store the set S'_i in a data structure TDS_i (of type TDS) for solving generalized queries of the form $TPR(q, [a, \infty), S'_i)$.

Having defined the transformed data structure, we show how to answer generalized queries of the form $TPR(q, [a, \infty), S)$. Without loss of generality we may assume that $a \leq k_{p_1}$ and $a > k_{p_n}$; otherwise, in the former case, the answer is the empty set while in the latter case there is effectively no range-restriction. Find the index i such that

$$k_{p_{im}} \geq a > k_{p_{(i+1)m}}.$$

Then, solve the generalized query $PR(q, S_i)$ using the structure DS_i, and solve the generalized query $TPR(q, [a, \infty), S'_i)$ using the structure TDS_i. Output the union of the colors reported by these two queries, after having removed duplicate colors.

Lemma 1 *The basic transformation results in a data structure for the generalized searching problem $TPR(q, [a : \infty), S)$*

1. *with a query time of $O((\log n)^{\max(u,v,1)} + C)$,*
2. *using $O(n^{2+\epsilon}/m + n\, m^{w-1})$ space.*

Proof: To prove the correctness of the query algorithm, observe that all elements p of $S \setminus (S_i \cup S'_i)$ have a parameter k_p that is smaller than a. Hence, these elements certainly do not satisfy the query. Moreover, all elements p of the set S_i have a parameter k_p that satisfies the range restriction.

The query time of the transformed data structure is upper-bounded by the $O(\log n)$ time to compute the index i plus the times to query DS_i and TDS_i. Hence, the total query time is $O((\log n)^{\max(u,v,1)} + C)$. (Note that each color is reported at most once by the query on DS_i and, similarly, once by the query on TDS_i.)

It remains to prove the space bound. The total size of the structures DS_i, $0 \leq i < n/m$, is bounded by

$$O\left(\sum_{i=0}^{n/m} (im)^{1+\epsilon}\right) = O\left(m^{1+\epsilon} \cdot (n/m)^{1+\epsilon} \cdot (n/m)\right) = O(n^{2+\epsilon}/m).$$

Similarly, the total size of the structures TDS_i is bounded by

$$O\left(\sum_{i=0}^{n/m} m^w\right) = O(n\, m^{w-1}).$$

This completes the proof. ■

Using this lemma, we can prove the main result of this section.

Theorem 1 *Let DS be a data structure that stores a set S of n colored objects, such that generalized queries $PR(q, S)$ can be solved in $O((\log n)^u + C)$ time, for some positive constant u. Let the size of DS be bounded by $O(n^{1+\epsilon})$, where ϵ is an arbitrarily small positive constant. Let TDS be a data structure for the set S, such that generalized queries $TPR(q, [a : \infty), S)$ can be solved in $O((\log n)^v + C)$ time, for some positive constant v. Let the size of TDS be bounded by $O(n^w)$ for some constant $w > 1$.*

There exists a data structure that solves generalized queries $TPR(q, [a : \infty), S)$

1. *with a query time of $O((\log n)^{\max(u,v,1)} + C)$,*
2. *using $O(n^{1+\epsilon})$ space, for an arbitrarily small positive constant ϵ.*

Proof: First assume that $w > 2$, i.e., the data structure TDS uses more than quadratic space. Choosing $m = n^{1/w}$ in Lemma 1 gives a data structure for solving queries $TPR(q, [a : \infty), S)$, with a query time of $O((\log n)^{\max(u,v,1)} + C)$, and using space

$$O(n^{2+\epsilon-1/w} + n^{2-1/w}),$$

which is bounded by $O(n^2)$, provided $\epsilon \leq 1/w$.

Hence, we can assume that the generalized problem TPR can be solved with quadratic space and $O((\log n)^z + C)$ query time, where $z = \max(u, v, 1)$.

We will prove by induction that for any positive integer constant ℓ there is a data structure TDS^ℓ for the generalized problem TPR, with $O((\log n)^z + C)$ query time, using $O(n^{1+1/\ell+\epsilon})$ space. Choosing ℓ large enough, the space bound becomes $O(n^{1+2\epsilon})$. Hence, starting with $\epsilon/2$ instead of ϵ will complete the proof.

For $\ell = 1$, the claim has been proved already. So, let $\ell \geq 1$, and assume the claim holds for ℓ. Apply Lemma 1 to the structures DS and TDS^ℓ, choosing $m = n^{\ell/(\ell+1)}$. This gives a data structure $TDS^{\ell+1}$ for solving queries $TPR(q, [a : \infty), S)$. Straightforward calculations show that $TDS^{\ell+1}$ has a query time of $O((\log n)^z + C)$, and uses $O(n^{1+1/(\ell+1)+\epsilon})$ space. ■

Corollary 1 *Let DS and TDS be as defined in Theorem 1. There exists a data structure that solves generalized queries* $TPR(q, [a : b], S)$

1. *with a query time of* $O((\log n)^{\max(u,v,1)} + C)$,
2. *using* $O(n^{1+\epsilon})$ *space, for an arbitrarily small positive constant* ϵ.

Proof: By Theorem 1, there exists a data structure TDS' that solves generalized queries $TPR(q, [a : \infty), S)$ with a query time of $O((\log n)^{\max(u,v,1)} + C)$, and using $O(n^{1+\epsilon})$ space. By a symmetric argument, there exists a data structure TDS'' that solves generalized queries $TPR(q, (-\infty, b], S)$ within the same time and space bounds.

We store the objects of S at the leaves of a balanced binary search tree, sorted in non-decreasing order of their parameter k_p. For each non-root node u of this tree, let S_u denote the subset of S that is stored at the leaves of u's subtree. If u is a left child, then this node u contains a pointer to a data structure TDS'_u storing S_u and that solves generalized queries $TPR(q, [a, \infty), S_u)$. Otherwise, if u is a right child, then u contains a pointer to a data structure TDS''_u storing S_u and that solves generalized queries $TPR(q, (-\infty, b], S_u)$.

It is easy to see that the entire data structure, i.e., the binary tree and the associated structures, has size $O(n^{1+\epsilon})$.

A query $TPR(q, [a : b], S)$ is solved as follows. Search in the tree for the values a and b. Let u be the node in which the two search paths diverge, and let u_l (resp. u_r) be the left (resp. right) child of u. Then we solve a generalized query $TPR(q, [a, \infty), S_{u_l})$ in the data structure TDS'_{u_l}, a generalized query $TPR(q, (-\infty, b], S_{u_r})$ in the data structure TDS''_{u_r}, and remove duplicate colors in the two answers. Clearly, the query time is bounded by $O((\log n)^{\max(u,v,1)} + C)$. ■

Theorem 1 and Corollary 1 imply that in order to solve the generalized problem TPR, it suffices to have (i) a data structure for PR with polylogarithmic query time and using roughly linear space, and (ii) a data structure for TPR with polylogarithmic query time and (possibly high) polynomial space. In many applications, the latter data structure is obtained from the following result, which is due to van Kreveld [8] (Corollary 7.3 (ii)). For convenience, we restate the result in [8] in a slightly different form.

Theorem 2 ([8]) *Let* S *be a set of* n *colored points in* $\mathbb{R}^d$, *where* $d \geq 2$ *is a constant, and let* k *be an integer constant. There exists a data structure of*

size $O(n^{d+\epsilon})$, such that for any query region q in $\mathbb{R}^d$ that is the intersection of at most k halfspaces, we can in $O(\log n + C)$ time report the C distinct colors of all points of S that are contained in q. Here ϵ is an arbitrarily small positive constant. ■

3 Generalized query problems for fat triangles

Let S be a set of n colored points in the plane. We want to store these points in a data structure, such that generalized *fat triangle* queries can be solved efficiently. In such a query, we are given a fat triangle, and we have to report the distinct colors of all points of S that are contained in the triangle. A triangle is called *fat*, if all its angles are at least equal to some fixed constant α.

We start by solving a simpler problem. Then, we apply Theorem 1 twice in order to get our result.

Let R be a fixed ray that starts in the origin. If q is a point in the plane, then R_q denotes the ray that is obtained by translating R such that its starting point coincides with q, i.e., $R_q = R + q$. Consider generalized queries of the following form: Given a point q and a ray T starting in q, such that T makes an angle at most π with R_q, we want to report the distinct colors of those points of S that are contained in the wedge defined by T and R_q. By Theorem 2, there is a data structure TDS for this wedge problem having $O(\log n + C)$ query time and using polynomial space.

Let DS be the data structure of [5] for solving the generalized halfplane range searching problem with $O((\log n)^2 + C)$ query time using $O(n \log n)$ space.

We show that we obtain the above wedge problem by adding a half-infinite range restriction to DS. In a query, we want to report the distinct colors of those points that are, say, below the line through T and, say, above the line through R_q. Let ℓ be the line through the origin orthogonal to R. Project all points of S onto ℓ. Each point of S gets this projection as the additional parameter. Note that these parameters define an ordering along ℓ in the natural way. A point of S is above the line through R_q if and only if its additional parameter is "larger" than the projection of q onto ℓ, where "larger" refers to the ordering along ℓ. Hence, among all points that are larger than q's projection, we want the distinct colors of those that are below the line through T.

Therefore, applying Theorem 1 to DS and TDS, we get the following

result.

Lemma 2 *Let S be a set of n colored points in the plane, and let R be a fixed ray that starts in the origin. There exists a data structure of size $O(n^{1+\epsilon})$ such that for any point q and any ray T starting in q, such that T makes an angle at most π with R_q (i.e., the ray R translated to q), we can report in $O((\log n)^2 + C)$ time, all C distinct colors of those points of S that are contained in the wedge defined by T and R_q. Here, ϵ is an arbitrarily small positive constant.* ■

Next, we show how to solve generalized *fat wedge* queries. In such a query, we are given a wedge whose angle is between α and π, where α is a constant.

Choose $t = \lceil 2\pi/\alpha \rceil$ coordinate systems $CS_i = (x_i, y_i)$, all sharing the origin, such that CS_{i+1} is offset from CS_i by an angle α. For each i, let DS_i be the data structure of Lemma 2 storing the points of S, where we take $R = x_i$.

A fat wedge query is solved as follows. Let q be the apex of the query wedge, and let A and B be its bounding rays. Find an index i, such that $(x_i)_q$—i.e., x_i translated by the vector q—is contained in the wedge. Then we query the data structure DS_i twice, once with the wedge defined by A and $(x_i)_q$, and once with the wedge defined by B and $(x_i)_q$. We output the union of the colors reported by these two queries, after having removed duplicate colors. This proves:

Lemma 3 *Let S be a set of n colored points in the plane. There exists a data structure of size $O(n^{1+\epsilon})$ such that for any fat query wedge, we can report in $O((\log n)^2 + C)$ time, all C distinct colors of those points of S that are contained in it. Here, ϵ is an arbitrarily small positive constant.* ■

Now we are ready to consider generalized fat triangle queries. Let T be a fat query triangle. We decompose T into two triangles T_1 and T_2 by drawing a vertical line through the middle vertex. Hence, for $i = 1, 2$, T_i has a vertical side, and the vertex opposite to this side has angle at least α. We can think of T_i as a range-restricted fat-wedge query, where the additional parameter of a point of S is its x-coordinate. Our goal is to get a data structure DS' for such a query which uses $O(n^{1+\epsilon})$ space and has a query time of $O((\log n)^2 + C)$. Given DS', we can solve the fat triangle query for T by querying DS' with T_1 and T_2, and reporting the union of the colors reported by these two queries.

Let DS be the data structure of Lemma 3. By Theorem 2, there is a data structure TDS for range-restricted fat-wedge queries having $O(\log n + C)$ query time and using polynomial space. Then, Theorem 1 gives the data structure DS'. This proves the following result.

Theorem 3 *Let S be a set of n colored points in the plane. There exists a data structure of size $O(n^{1+\epsilon})$ such that for any fat query triangle, we can report in $O((\log n)^2 + C)$ time, all C distinct colors of those points of S that are contained in it. Here, ϵ is an arbitrarily small positive constant.* ■

Using basically the same approach, we can solve the generalized searching problem of querying fat triangles with points. We leave the details to the reader.

Theorem 4 *Let S be a set of n colored fat triangles in the plane. There exists a data structure of size $O(n^{1+\epsilon})$ such that for any query point q, we can report in $O((\log n)^2 + C)$ time, all C distinct colors of those triangles of S that contained q. Here, ϵ is an arbitrarily small positive constant.* ■

4 Concluding remarks

We have given a general technique for adding a range restriction to a generalized searching problem. This results in data structures for generalized searching problems on fat triangles having $O((\log n)^2 + C)$ query time using $O(n^{1+\epsilon})$ space.

Our technique can also be used to solve the d-dimensional generalized orthogonal range searching problem with $O(\log n + C)$ query time and $O(n^{1+\epsilon})$ space. (This was known already, see Corollary 7.3 (i) in [8].) It remains open if this problem can be solved with $O(\text{polylog}(n) + C)$ query time and $O(n(\log n)^{O(1)}n)$ space for dimensions $d \geq 4$. (For efficient solutions in dimensions $d \leq 3$, see [4, 6].)

The results of this paper only apply to generalized *reporting* problems. In a generalized *counting* problem, we want to report the *number* of distinct colors of the objects that intersect the query object. Is there a general technique for adding a range restriction to such generalized counting problems?

Acknowledgement

The authors would like to thank the referees for several suggestions that helped improve the presentation.

References

[1] P.K. Agarwal and M. van Kreveld. *Polygon and connected component intersection searching.* Algorithmica **15** (1996), 1993, pp. 626–660.

[2] J.L. Bentley. *Decomposable searching problems.* Information Processing Letters **8** (1979), pp. 244–251.

[3] P. Gupta, R. Janardan, and M. Smid. *On intersection searching problems involving curved objects.* Proceedings 4th Scandinavian Workshop on Algorithm Theory, Lecture Notes in Computer Science, Vol. 824, Springer-Verlag, Berlin, 1994, pp. 183–194.

[4] P. Gupta, R. Janardan, and M. Smid. *Further results on generalized intersection searching problems: counting, reporting, and dynamization.* Journal of Algorithms **19** (1995), pp. 282–317.

[5] P. Gupta, R. Janardan, and M. Smid. *Algorithms for generalized halfspace range searching and other intersection searching problems.* Computational Geometry: Theory and Applications **5** (1996), pp. 321–340.

[6] R. Janardan and M. Lopez. *Generalized intersection searching problems.* International Journal on Computational Geometry & Applications **3** (1993), pp. 39–69.

[7] M. Katz. *3-D vertical ray shooting and 2-D point enclosure, range searching, and arc shooting amidst convex fat objects.* Report INRIA, Nr. 2583, 1995.

[8] M. van Kreveld. *New results on data structures in computational geometry.* Ph.D. Thesis, Department of Computer Science, University of Utrecht, the Netherlands, 1992.

[9] H.-P. Lenhof and M. Smid. *Using persistent data structures for adding range restrictions to searching problems.* RAIRO Theoretical Informatics and Applications **28** (1994), pp. 25–49.

[10] H.W. Scholten and M.H. Overmars. *General methods for adding range restrictions to decomposable searching problems.* Journal of Symbolic Computation **7** (1989), pp. 1–10.

[11] D.E. Willard and G.S. Lueker. *Adding range restriction capability to dynamic data structures.* Journal of the ACM **32** (1985), pp. 597–617.