Parallel Algorithms for Grounded Range Search and Applications

Michael G. Lamoureux and Andrew Rau-Chaplin

Faculty of Computer Science
Dalhousie University
P.O. Box 1000, Halifax, Canada B3J 2X4.
Voice: 902-494-2732, Fax 902-492-1517.
mlamour@cs.dal.ca, arc@cs.dal.ca

Abstract. This paper presents a parallel algorithm for solving grounded range search in associative-function mode using the BSP-like Coarse Grained Multicomputer (CGM). Given a set S of n weighted points in the plane, the algorithm requires $O(1)$ communication rounds (h-relations with $h = O(n/p)$), $O((n/p)\log n)$ local computation, and $O(n/p)$ memory per processor ($n/p \geq p$), to solve $m = O(n)$ grounded range search problems. The result implies new or improved solutions to a number of other geometric applications including d-dimensional range search, quadrant search, interval intersection, and chromatic range queries.

1 Introduction

A *grounded range search query* in the plane is a 2-dimensional domain given by the 4-tuple $(x_{min}, x_{max}, -\infty, y_{max})$ which specifies a geometric query range. Let S be a set of n points in the plane with some *weight* $w(v)$ assigned to each $v \in S$, let Q be a set of $m = O(n)$ grounded range search queries, and let $\otimes$ be a binary associative function applied to a subset of weighted points.

The *2D grounded range search* problem consists of determining for each $q \in Q$ either the subset of the points in S contained in the domain of q, or the number of such points, or more generally $result(q, S)$, the result of applying the binary associative function $\otimes$ over such points. The former version of this problem is called the *report mode* while the latter versions are called the *counting* and *associative-function modes*, respectively.

The classical sequential solution to this problem in counting mode combines the results of two dominance queries which can be answered in $O(\log n)$ time and $O(n)$ space using the method of [2]. This reduction from grounded range search to dominance is applicable in this case because addition is not only associative and commutative but also cancellative (i.e. $x \otimes a = y \otimes a \Rightarrow x = y$). However, for many useful non-cancellative operators, like Max, this approach is not applicable.

The classical sequential solution to this problem in associative-function and reporting modes uses the $O(n)$ size priority search tree of McCreight [21] which answers a grounded range search query in $O(\log n)$ and $O(\log n + t)$ time, respectively, where t is the size of the output.

P. Amestoy et al. (Eds.): Euro-Par'99, LNCS 1685, pp. 525–532, 1999.

This paper presents an efficient parallel algorithm for solving the associative-function mode variant of the grounded range search problem for a set of $m = O(n)$ queries and n weighted points in the plane on a p-processor coarse grained multicomputer with arbitrary interconnection network and local memories of size $O(\frac{n}{p})$, $\frac{n}{p} \geq p$, in time $O(\frac{n \log n}{p} + T_h(n,p))$, where $T_h(n,p)$ is the time required to route a single h-relation with $h = O(n/p)$ [22, 23]. Note that the local computation time of this algorithm is optimal and that it requires only a constant number of communication rounds.

This algorithm permits efficient parallel solutions to d-dimensional range search, quadrant search, interval intersection search, and chromatic range search.

This paper reduces the query time required for $m = O(n)$ d-dimensional range queries from $O(\frac{n \log^d n}{p} + T_h(s,p))$ to $O(\frac{n \log^{d-1} n}{p} + T_h(s,p))$ in associative-function mode without increasing the storage requirement, thus improving upon the solution of [14] by a $\log n$ factor in search time.

This paper also presents algorithms to solve $m = O(n)$ associate-function mode quadrant search, interval intersection search, or chromatic range search queries on a set of n weighted points in the plane using a p-processor coarse grained multicomputer with arbitrary interconnection network and local memories of size $O(\frac{n}{p})$, $\frac{n}{p} \geq p$, in time $O(\frac{n \log n}{p} + T_h(n,p))$. These are, to the best of our knowledge, the first coarse grained parallel solutions to these problems.

2 The Coarse Grained Multicomputer Model

We are using a variation of the BSP model, referred to as the *Coarse Grained Multicomputer*, CGM [4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15]. It is comprised of a set of p processors $P_1, \ldots, P_p$ with $O(n/p)$ local memory per processor and an arbitrary communication network (or shared memory). The term "coarse grained" refers to the fact that we assume that the size $O(n/p)$ of each local memory is "considerably larger" than $O(1)$. Our definition of "considerably larger" will be that $n/p \geq p$.

All algorithms consist of alternating local computation and global communication rounds. Each communication round consists of routing a single h-relation with $h = O(n/p)$, i.e. each processor sends $O(n/p)$ data and receives $O(n/p)$ data. We require that all information sent from a given processor to another processor in one communication round is packed into one message. In the BSP model, a computation/communication round is equivalent to a superstep with $L = (n/p)g$ (plus the above "packing" and "coarse grained" requirement).

Finding an optimal algorithm in the coarse grained multicomputer model is equivalent to minimizing the number of communication rounds as well as the total local computation time. This considers all parameters discussed above that are affecting the final observed speedup, and it requires no assumption on g. Furthermore, it has been shown that minimizing the number of supersteps also leads to improved portability across different parallel architectures [9, 22, 23]. The above model has been used in parallel algorithm design for various problems ([3, 5, 6, 7, 8, 10, 13, 20]) and has demonstrated good practical timing results.

We now list some operations required by our algorithms. Each of these operations reduces to $O(1)$ communication rounds for $n/p \geq p$.

All-to-all broadcast: Every processor sends one message to all other processors [6] ($O((n/p))$ local computation).

Personalized all-to-all broadcast: Every processor (in parallel) sends a different message to every other processor [6] ($O((n/p))$ local computation).

Partial sum (Scan): Every processor stores one value, and all processors compute the partial sums of these values with respect to some associative operator [6] ($O((n/p))$ local computation).

Global sort: Sort $O(n)$ data items stored on a CGM, n/p data items per processor, with respect to the CGM's processor numbering. As shown in [17], for $n/p \geq p$ it is possible to sort in $O(1)$ communication rounds with $O(n)$ memory per processor and $O((n/p)\log n)$ local computation.

Global integer sort: Sort $O(n)$ integers in the range $1, \ldots, n^c$ for fixed constant c stored on a CGM, n/p data items per processor, with respect to the CGM's processor numbering. The sort algorithm in [17] is based on Cole's merge sort [16]. The $O((n/p)\log n)$ local computation in [17] is due to a constant number of local sorts. Hence, by applying radix sort for the integer case, we obtain $O(n/p)$ local computation without increasing the number of communication rounds. A CGM integer sorting algorithm with 9 communication rounds, $O(n/p)$ memory per processor and $O(n/p)$ local computation can be found in [4].

Load Balance: Given a set $\bar{Q}$ of $m = O(n)$ queries where associated with each query is a value $next(q)$ which is the name of the substructure it next requires and a distributed data structure $\bar{S}$ which consists of p substructures $\bar{S}_i$ ($1 \leq i \leq p$) of size $O(n/p)$ stored with $\bar{S}_i$ on processor p_i of a CGM(n,p). This operation balances queries and structures such that each query $q \in \bar{Q}$ is stored on a processor which also stores a copy of the substructure $next(q)$.

Algorithm 1 "Load Balance($\bar{S}, \bar{Q}$)".

Architecture: A p-processor coarse grained multicomputer, $CGM(n,p)$, with arbitrary interconnection network and local memories of size $O(\frac{n}{p})$, $\frac{n}{p} \geq p$.

Input: Each processor p_i stores a set $\bar{Q}_i$ of n/p queries from $\bar{Q}$ and a substructure $\bar{S}_i$ of size $O(n/p)$ from $\bar{S}$. Associated with each query is the value $next(q)$ which is the name of the substructure it next requires.

Output: Each query $q \in \bar{Q}$ is stored on a processor which also stores a copy of the substructure $next(q)$.

1 Globally compute $c(\bar{S}_i) = \lceil \frac{|\{q \in \bar{Q} | next(q) = i\}|}{\frac{n}{p}} \rceil$

2 Make $c(\bar{S}_i)$ copies of S_i and distribute them evenly such that each processor stores at most two substructures.

3 Redistribute $\bar{Q}$ evenly so that every query $q \in \bar{Q}$ is stored on a processor that also stores a copy of the element of $\bar{S}$ which q is visiting.

— End of Algorithm —

Note that this algorithm evenly distributes queries and substructures $\bar{S}_i$, such that each processor has $O(1)$ copies of each. This approach to load balancing is based on a CGM technique described and analyzed in [6] and parallel integer sorting, it requires $O(1)$ communication rounds and $O(n/p)$ local computation.

3 A Parallel Algorithm for Grounded Range Search

Consider a set of p horizontal lines h_i which partition S into p subsets H_i of $\frac{n}{p}$ points each (with h_i below H_i, and h_{i+1} above H_i). Analogously, consider p vertical lines l_j which partition S into p subsets V_j of $\frac{n}{p}$ points each (with l_j to the left of V_j, and l_{j+1} to the right of V_j). For a subset $A \subset S$ let $w(A) = \sum_{a \in A} w(a)$. Let V_{ij} be the set of points of V_j that are below h_i. Let H and V be the set of points in H_i $(1 \leq i \leq p)$ and V_j $(1 \leq j \leq p)$, respectively which are assumed to be in general position without loss of generality. (See Figure 1.)

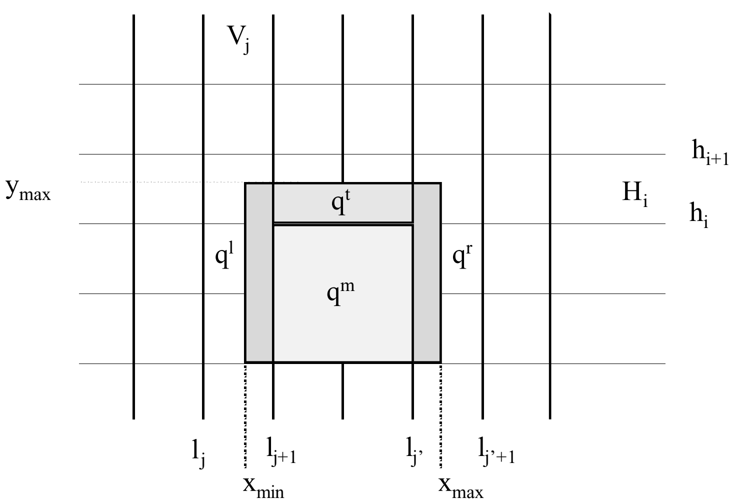

Fig. 1. A grounded range search query with respect to a set of horizontal and vertical partitions of the the plane.

Observation 1 *For each query* $q \in Q$ *with* $x_{min} \in [l_j, l_{j+1}]$, $x_{max} \in [l_{j'}, l_{j'+1}]$,*and* $y_{max} \in [h_i, h_{i+1}]$ *let* $q^l = (x_{min}, l_{j+1}, -\infty, y_{max})$, $q^r = (l_{j'}, x_{max}, -\infty, y_{max})$, $q^t = (l_{j+1}, l_{j'}, h_i, y_{max})$, *and* $q^m = (l_{j+1}, l_{j'}, -\infty, h_i)$. *Note that* $result(q, S) = result(q^l, V_j) \otimes result(q^r, V_{j'}) \otimes result(q^t, H_i) \otimes result(q^m, \cup_{k=j+1}^{j'-1} V_{ik})$.

Algorithm 2 "Grounded Range Search".
Architecture: A p-processor coarse grained multicomputer, $CGM(n,p)$, with arbitrary interconnection network and local memories of size $O(\frac{n}{p})$, $\frac{n}{p} \geq p$.

Input: Each processor stores $\frac{n}{p}$ points of S.
Output: Each processor stores $result(q, S)$ for each of its $\frac{n}{p}$ queries $q \in Q$.

1 Globally sort S by x-coordinates such that processor p_j stores V_j and l_j. Perform an all-to-all broadcast, where processor p_j sends l_j to all other processors. Every processor now stores all vertical lines $l_1, \ldots, l_p$.

2 Each processor p_j uses $l_1, \ldots, l_p$ to constructs subqueries q^l and q^r and computes $next(q^l)$ and $next(q^r)$ for each query q it stores. Let Q^{lr} denote the set of all q^l and q^r queries.

3 Call Load-Balance(V,Q^{lr}) and then solve all queries in Q^{lr} sequentially associating the result with the query.

4 Globally sort a copy of S by y-coordinates such that processor p_i stores H_i and h_i. Perform an all-to-all broadcast, where processor p_i sends h_i to all other processors. Every processor stores now all horizontal lines $h_1, \ldots, h_p$.

5 Each processor p_j which stores V_j uses horizontal lines $h_1, \ldots, h_p$ to construct q^t and q^m and compute $next(q^t)$ and $next(q^m)$ for each query q it stores. Let Q^{tm} denote the set of all q^t and q^m queries. Each processor also computes $w(V_{ij})$ for $i \in h_1, \ldots, h_p$. Perform an all-to-all broadcast, where processor p_j sends $w(V_{ij})$ to processor p_{i+1}, $1 \leq i < p$.

6 Each processor p_i which stores H_i and a part of Q^{tm} now also stores $V_{i-1,j}(1 \leq j \leq p)$ associated with H_i. Load-Balance(H,Q^{tm}) and solve the q^m queries by performing a partial sum in $V_{i-1,j}(1 \leq j \leq p)$ and the q^t queries by sequential grounded range search. Associate the results with the query.

7 Globally sort $Q^{lr} \cup Q^{tm}$ by query index such that for each original query q the subqueries q^l, q^r, q^t, q^m are contiguous. Use a scan operation with $\otimes$ to compute the final result for each original query.

— End of Algorithm —

Theorem 1. *The grounded range search problem in associative-function mode for a set of $m = O(n)$ queries and n weighted points in the plane can be solved on a p-processor coarse grained multicomputer with arbitrary interconnection network and local memories of size $O(\frac{n}{p})$, $\frac{n}{p} \geq p$, in time $O(\frac{n \log n}{p} + T_h(n,p))$, where $T_h(n,p)$ is the time required to route a single h-relation with $h = O(n/p)$.*

Proof. The correctness of Algorithm 2 follows from Observation 1. Steps 1-3 solve the q^l and q^r queries in $O(\frac{n \log n}{p})$ local computation (from sorting and sequential grounded range search) and $O(1)$ communications rounds (from sorting, load-balancing, the distribution of vertical cutting lines). Since only $O(1)$ copies of points and queries are made and there are at most p cutting lines the space requirement is $O(\frac{n}{p} + p) = O(\frac{n}{p})$ per processor. Steps 4-6 solve the q^t and

q^m queries in $O(\frac{n \log n}{p})$ local computation (again from sorting and sequential grounded range search) and $O(1)$ communications rounds (from sorting, load-balancing, and the distribution of $w(V_{ij})$ and the horizontal cutting lines). Again, since only $O(1)$ copies of points and queries are made, and since there are at most p cutting lines, the space requirement is $O(\frac{n}{p} + p) = O(\frac{n}{p})$ per processor.

The local computation time for each step is $O(\frac{n}{p} \log n)$. The global communication in each step reduces to a constant number of global sorts and communication operations listed in Section 2 and the time complexity follows. □

4 Applications

This section describes efficient parallel solutions to d-dimensional range queries, quadrant queries, interval intersection queries, and chromatic range queries using grounded range search.

The authors of [14] demonstrate how to construct a distributed range tree T on a d-dimensional set S of n points on a coarse grained multicomputer in $O(\frac{n \log^{d-1} n}{p} + T_h(s,p))$ time to answer a set Q of $m = O(n)$ range queries in time $O(\frac{n \log^{d} n}{p} + T_h(s,p))$ in associative-function mode. If we use grounded range search queries in the last two structural dimensions, we can reduce the query time by a $\Theta(logn)$ factor to $O(\frac{n \log^{d-1} n}{p} + T_h(s,p))$ in associative-function mode.

Theorem 2. *The d-dimensional range search problem in associative-function mode for a set of $m = O(n)$ queries and n weighted points in d-space can be solved on a p-processor coarse grained multicomputer with arbitrary interconnection network and local memories of size $O(\frac{n \log^{d-1} n}{p})$, $\frac{n \log^{d-1} n}{p} \geq p$, in time $O(\frac{n \log^{d-1} n}{p} + T_h(n,p))$, where $T_h(n,p)$ is the time required to route a single h-relation with $h = O(\frac{n \log^{d-1} n}{p})$.*

Proof. If we use the algorithm of [14] to build a modified d-dimensional range tree where the structures in the dth dimension are simply stored as point sets, we can solve a d-dimensional range query by solving a (d-2)-dimensional range query on the first (d-2) dimensions of the d-dimensional structure and solving two grounded range search queries on the point sets associated with the left and right children of each node in range in the dimension (d-1) substructures using the method of Edelsbrunner [12]. □

Given a point v=(x,y) in the plane, a quadrant query asks for all points that lie in one of the four quadrants defined by the point. Since the quadrants are defined by semi-infinite ranges in the x and y directions, a quadrant query may be viewed as a grounded range search query with one side open.

Theorem 3. *The quadrant search problem in associative-function mode for a set of $m = O(n)$ queries and n weighted points in the plane can be solved on a p-processor coarse grained multicomputer with arbitrary interconnection network and local memories of size $O(\frac{n}{p})$, $\frac{n}{p} \geq p$, in time $O(\frac{n \log n}{p} + T_h(n,p))$, where $T_h(n,p)$ is the time required to route a single h-relation with $h = O(n/p)$.*

Proof. Omitted.

Given a set S of n weighted line segments, the associative interval intersection problem asks for the result of a binary associative operator applied to the weights of each pair of intervals in the set S that intersect. Grounded range search may be used to construct a very elegant solution to this problem. If we follow the precedent of McCreight [21] and map the intervals $[a, b]$ in S to the points (a, b) in S' and the query intervals $[u, v]$ in Q to the points (u, v) in Q', we can solve our interval intersection queries by performing a quadrant search on $[u, \infty) * (-\infty, v]$ for each query point $q' = (u, v)$ in Q.

Theorem 4. *The interval intersection search problem in associative-function mode for a set of $m = O(n)$ queries and n weighted points in the plane can be solved on a p-processor coarse grained multicomputer with arbitrary interconnection network and local memories of size $O(\frac{n}{p})$, $\frac{n}{p} \geq p$, in time $O(\frac{n \log n}{p} + T_h(n, p))$, where $T_h(n, p)$ is the time required to route a single h-relation with $h = O(n/p)$.*

Proof. Omitted.

Janardan and Lopez [19] define a chromatic query as a query on a set S of n geometric objects which belong to g disjoint groups, each labeled with a color, and it is the groups, and not the objects, which are of interest. In associative mode, the groups are weighted and a chromatic range query is a range query which asks for the result of the repeated application of a binary associative operator to each group that contains a datapoint located in the given range.

Theorem 5. *The chromatic search problem in associative-function mode for a set of $m = O(n)$ queries and n weighted points in the plane can be solved on a p-processor coarse grained multicomputer with arbitrary interconnection network and local memories of size $O(\frac{n}{p})$, $\frac{n}{p} \geq p$, in time $O(\frac{n \log n}{p} + T_h(n, p))$, where $T_h(n, p)$ is the time required to route a single h-relation with $h = O(n/p)$.*

Proof. If we use the technique of Gupta, Janardan, and Smid [18] and transform each point p in S to the point $p' = (p, pred(p))$ in S' where $pred(p)$ is its immediate predecessor of the same color (or $-\infty$ if the point p has no predecessor). Note that this transformation is such that there is a point p of color c in the query interval $q = [l, r]$ if and only if there is a point p' of color c in the grounded query rectangle $q' = [l, r] * (-\infty, l)$. Thus, we can use a grounded range search on the transformed point set S'. □

References

[1] S.G. Akl and K.A. Lyons, "Parallel Computational Geometry", Prentice Hall, 1993.

[2] J.L. Bentley, "Multidimensional Divide and Conquer", Comm. ACM 23(4):214-229, 1980.

[3] G.E. Blelloch, C.E. Leiserson, B.M. Maggs and C.G. Plaxton, "A Comparison of Sorting Algorithms for the Connection Machine CM-2", in Proc. ACM Symp. on Parallel Algorithms and Architectures, 1991, pp. 3-16.
[4] A. Chan, F. Dehne, "A Note on Coarse Grained Parallel Integer Sorting," Technical Report TR-98-06, School of Computer Science, Carleton University.
[5] F. Dehne, A. Fabri and C. Kenyon, "Scalable and Architecture Independent Parallel Geometric Algorithms with High Probability Optimal Time", in Proc. 6th IEEE Symp. Parallel and Distributed Processing, 1994, pp. 586-593.
[6] F. Dehne, A. Fabri and A. Rau-Chaplin, "Scalable Parallel Computational Geometry for Coarse Grained Multicomputers", in Proc. ACM Symp. Computational Geometry, 1993, pp. 298-307.
[7] F. Dehne, X. Deng, P. Dymond, A. Fabri and A.A. Kokhar, "A randomized parallel 3D convex hull algorithm for coarse grained parallel multicomputers", in Proc. ACM Symp. on Parallel Algorithms and Architectures, 1995.
[8] X. Deng, "A Convex Hull Algorithm for Coarse Grained Multiprocessors", in Proc. 5th Int. Symp. on Algorithms and Computation, 1994.
[9] X. Deng and P. Dymond, "Efficient Routing and Message Bounds for Optimal Parallel Algorithms", in Proc. Int. Parallel Proc. Symp., 1995.
[10] X. Deng and N. Gu, "Good Programming Style on Multiprocessors", in Proc. IEEE Symp. on Parallel and Distributed Processing, 1994, pp. 538-543.
[11] O. Develliers and A. Fabri, "Scalable Algorithms for Bichromatic Line Segment Intersection Problems on Coarse Grained Multicomputers", (WADS'93), Springer Lecture Notes in Computer Science, 1993, pp. 277-288.
[12] H. Edelsbrunner, "A Note on Dynamic Range Searching", Bulletin of the EATCS 15:34-40, 1981.
[13] A. Ferreira, A. Rau-Chaplin, S. Ubeda, "Scalable 2D Convex Hull and Triangulation for Coarse Grained Multicomputers", in Proc. 6th IEEE Symp. on Parallel and Distributed Processing, San Antonio, 1996.
[14] A. Ferreira, C. Kenyon, A. Rau-Chaplin, S. Ubeda, "d-Dimensional Range Search on Multicomputers", Proc. 11th Int. Parallel Processing Symposium (IPPS'97), April 1997, pp. 616-620.
[15] A.V. Gerbessiotis and L.G. Valiant, "Direct Bulk-Synchronous Parallel Algorithms", in Proc. 3rd Scandinavian Workshop on Algorithm Theory, Lecture Notes in Computer Science, Vol. 621, 1992, pp. 1-18.
[16] R. Cole, "Parallel Merge Sort", SIAM J. Comp. 17(4):770-785, 1998.
[17] M.T. Goodrich, "Communication Efficient Parallel Sorting", in Proc. 28th ACM Symp. on Theory of Computing (STOC'96), 1996.
[18] P. Gupta, R. Janardan, M. Smid, "Further results on generalized intsersection searching problems: counting, reporting, dynamization", J. Algs 19:282-317, 1995.
[19] Janardan, Ravi and Mario Lopez, "Generalized Intersection Searching Problems", Int. J. Comp. Geom. and Applications 3(1):39-69, 1993.
[20] H. Li and K.C. Sevcik, "Parallel Sorting by Overpartitioning", in Proc. ACM Symp. on Parallel Algorithms and Architectures, 1994, pp. 46-56.
[21] McCreight, E.M., "Priority Search Trees" SIAM J. Comput. 14(2):257-276, 1985.
[22] L.G. Valiant, "A Bridging Model for Parallel Computation", Comm. ACM 33:103-111, 1990.
[23] L.G. Valiant et. al., "General Purpose Parallel Architectures", Handbook of Theoretical Computer Science, Edited by J. van Leeuwen, MIT Press/Elsevier, 1990, pp. 943-972.