数理解析研究所講究録
950巻 1996年 120-125

An Optimal Algorithm for the Angle-Restricted All Nearest Neighbor Problem on the Reconfigurable Mesh †

Koji Nakano‡
Department of Electrical and Computer Engineering
Nagoya Institute of Technology
Showa-ku, Nagoya 466, Japan

Stephan Olariu
Department of Computer Science
Old Dominion University
Norfolk, Virginia 23529, USA

Abstract

Given a set S of n points in the plane and two directions r_1 and r_2, the Angle-Restricted All Nearest Neighbor problem (ARANN, for short) asks to compute for every point p in S the nearest point in S lying in the planar region bounded by two rays in the directions r_1 and r_2 emanating from p. The ARANN problem generalizes the well-known ANN problem and finds applications to pattern recognition, image processing, and computational morphology. Our main contribution is to present an algorithm that solves an arbitrary instance of size n of the ARANN problem in $O(1)$ time on a reconfigurable mesh of size $n \times n$. Our algorithm is optimal in the sense that $\Omega(n^2)$ processors are necessary to solve the ARANN problem in $O(1)$ time.

1 Introduction

A reconfigurable mesh [4] (RMESH, for short) of size $m \times n$ consists of nm identical SIMD processors positioned on a rectangular array with m rows and n columns. Each processor is connected to its four neighbors, provided they exist, and has 4 ports denoted by N, S, E, and W. Local connections between these ports can be established, under program control, creating a powerful bus system that changes dynamically to accommodate various computational needs. We assume that communications along buses take O(1) time. The results in this paper assume a model that allows at most two connections to be set in each processor at any one time. Furthermore, these two connections must involve disjoint pairs of ports.

The All-Nearest Neighbor problem (ANN, for short) involves computing for every point in a given set S, a point that is closest to it. Recently, Jang and Prasanna [1] provided an O(1) time algorithm for solving the ANN problem for n points in the plane on a RMESH of size $n \times n$. In this paper we address a generalization of the ANN problem, namely the the Angle-Restricted All Nearest Neighbor problem (ARANN, for short). Just as the ANN problem, the ARANN problem has wide-ranging applications in pattern recognition and image processing.

For points p and q in the plane, we let $d(p,q)$ stand for the Euclidean distance between p and q. Further, we say that q is (r_1,r_2)-*dominated* by p if q lies inside of the closed planar region determined by two rays in directions r_1 and r_2 emanating from p. In this terminology, a point q in S is said to be the (r_1,r_2)-nearest neighbor of p if q is (r_1,r_2)-dominated by p and $d(p,q) = \min\{d(p,q') \mid q' \in S$ and q' is (r_1,r_2)-dominated by $p\}$.
The ARANN problem involves determining the (r_1,r_2)-nearest neighbor of every point in S. Refer to Figure 1 for an illustration. Here, p_2 is $(0,\pi/3)$-dominated by p_1, but is not $(0,\pi/3)$-dominated by p_4. The $(0,\pi/3)$-nearest neighbor of p_1 is p_2.

Given set S of n points in the plane and two directions r_1 and r_2, the *Angle-Restricted All Nearest Neighbor* graph of S, denoted ARANN(S) is the directed graph whose vertices are the points in S; the points p and q are linked by a directed edge from p to q whenever q is the (r_1,r_2)-nearest neighbor of p. The reader will not fail to note that the problem of computing the (r_1,r_2)-closest

‡中野浩嗣, 名古屋工業大学電気情報工学科

†Work supported in part by NSF grant CCR-9407180 and by ONR grant N00014-95-1-0779

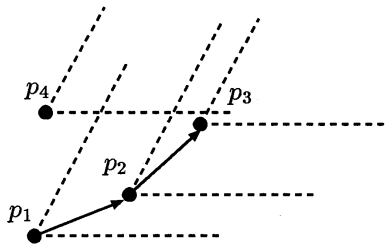

Figure 1: *Illustrating* $(0, \frac{\pi}{3})$*-domination and the corresponding ARANN graph*

neighbor of each point in S and the problem of computing the graph ARANN(S) of S are intimately related, in the sense that the solution to either of them immediately yields a solution to the other. For this reason in the remaining part of this work we shall focus on the problem of computing the graph ARANN(S) and we shall refer to this task, informally, as solving the ARANN problem.

The main contribution of this work is to present an algorithm to solve the ARANN problem of n points in $O(1)$ time on a RMESH of size $n \times n$. We also show that our algorithm is optimal in the sense that n^2 processors are necessary to solve the ARANN of n points in $O(1)$ time. Clearly, the ARANN problem is at least as hard as ANN: the ANN corresponds to the solution of ARANN for the particular directions 0 and 2π. It is worth noting that the ANN algorithm of Jang and Prasanna [1] cannot be used for solving the ARANN problem. We will develop new tools for dealing with the ARANN problem. These tools are interesting in their own right and may be of import in the resolution of other related problems. We omit the details but mention that our ARANN algorithm allows us to solve the problems of computing the Geographic Neighborhood Graph and the Relative Neighborhood Graph in $O(1)$ time on an $n \times n$ RMESH and that of the Euclidean Minimum Spanning Tree [7] in $O(1)$ time on an $n \times n^2$ RMESH. We will show algorithms for these problems in the full version of this paper.

2 Lower Bound

Let us consider the ARANN problem for the directions $-\pi/2$ and $\pi/2$. Consider a set $S = \{(a_1, 0), (a_2, 0), \ldots, (a_n, 0)\}$ of points on the x-axis such that point $(a_i, 0)$, $(1 \leq i \leq n)$, is assigned to the i-th column of an RMESH of size $m \times n$. After solving the ARANN problem, the processors of the i-th column know the $(-\pi/2, \pi/2)$-nearest neighbor of $(a_i, 0)$. Assume that $a_1 < a_{n/2} < a_2 < a_{n/2+1} < a_3 < a_{n/2+2} \cdots < a_{n/2-1} < a_n$. Then, for each point $(a_i, 0)$, $(1 \leq i \leq n/2 - 1)$, its $(-\pi/2, \pi/2)$-nearest neighbor is the point $(a_{i+n/2}, 0)$. Therefore, information about $n/2$ points $(a_{n/2}, 0), (a_{n/2+1}, 0), \ldots, (a_{n-1}, 0)$ must be transferred through the m links that connect the $(n/2-1)$-th column and the $n/2$-th column of the RMESH. Hence, $\Omega(n/m)$ time is required to solve the ARANN problem. We proved the following result.

Theorem 2.1 $\Omega(n^2)$ *processors are necessary to solve an instance of size n of the ARANN problem on the RMESH in* $O(1)$ *time.* □

3 An Optimal Algorithm

Consider a collection S of n points in the plane. By rotating the input about the origin, if necessary, we can assume that two directions are 0 and r with $0 \leq r < 2\pi$. The main goal of this section is to present an optimal $O(1)$-time algorithm for solving the ARANN problem. We begin by showing an efficient algorithm for computing the prefix-minima.

By comparing all pairs of n numbers, their minimum can be computed in $O(1)$ time on an $n \times n$ RMESH. Hence, the prefix-minima of n numbers can be computed in $O(1)$ time on an $n \times n^2$ RMESH. We will show the size of the RMESH can be reduced to $n \times n^\epsilon$ for every fixed $\epsilon > 0$. Partition the sequence $A = \{a_1, a_2, \ldots, a_n\}$ of n numbers into $n^{\epsilon/2}$ sequences $A_1, A_2, \ldots, A_{n^{\epsilon/2}}$ each of which contains $n^{1-\epsilon/2}$ numbers. Next, compute the (local) prefix-minima of each A_i $(1 \leq i \leq n^{\epsilon/2})$ on an $n^{1-\epsilon/2} \times n^\epsilon$ submesh recursively. Let $\max(A_i)$ be the minimum within A_i. Further, compute the (global) prefix-minima of a sequence $\{\min(A_1), \min(A_2), \ldots, \min(A_{n^{\epsilon/2}})\}$. This can be done in $O(1)$ time by the O(1) time algorithm discussed above. Finally, for each number $a_j (\in A_i)$, compute the minimum of its local prefix-minima and the global prefix-minima $\min\{A_1, A_2, \ldots, A_{i-1}\}$ that corresponds to the prefix-minima of a_j. Since the depth of the recursion is $O(1/\epsilon)$, the prefix-minima can be computed in $O(1)$ time. Thus, we have

Lemma 3.1 *Given n numbers on an $n \times n^{\epsilon}$ RMESH, their prefix-minima can be computed in $O(1)$ time for every fixed $\epsilon > 0$.* □

From Lemma 3.1 we can get a trivial suboptimal solution to the problem at hand.

Lemma 3.2 *For every fixed $\epsilon > 0$, an arbitrary instance of size n of the ARANN problem can be solved in $O(1)$ time on a RMESH of size $n \times n^{1+\epsilon}$.* □

In the remainder of this section we will show how to improve this naive algorithm to run in $O(1)$-time on a RMESH of size $n \times n$. First, assume that the given directions are 0 and r, with $0 < r < \pi/2$, that is, the angle of the closed region is acute. Consider a set $S = \{p_1, p_2, \ldots, p_n\}$ of points in the plane stored one per processor in the first row of a RMESH of size $n \times n$. The details of our algorithm follow.

Step 1. Sort the points in S by y-coordinate and partition them into $n^{1/3}$ subsets $Y_1, Y_2, \ldots Y_{n^{1/3}}$ of $n^{2/3}$ points each, such that the y-coordinate of all points in Y_i is smaller than the y-coordinate of all points in Y_{i+1};

Step 2. For each point p in S compute $x'(p) = x(p) - y(p)/\tan(r)$ and sort the points by x', where $x(p)$ and $y(p)$ are x- and y-coordinate of p. Next, partition the points into subsets $X_1, X_2, \ldots, X_{n^{1/3}}$ of $n^{2/3}$ points each, such that for every choice of points p in X_i and q in X_{i+1}, $x'(p) < x'(q)$;

Step 3. For each point p in X_i, $(1 \leq i \leq n^{1/3})$, find its $(0, r)$-nearest neighbor $X(p)$ in X_i;

Step 4. For each point p in Y_i, $(1 \leq i \leq n^{1/3})$, find its $(0, r)$-nearest neighbor $Y(p)$ in Y_i;

Step 5. For each i and j, $(1 \leq i, j \leq n^{1/3} - 1)$, let $Z_{i,j} = \cup_{i' > i, j' > j}(X_{i'} \cap Y_{j'})$. For each point p in $X_i \cap Y_j$ compute its $(0, r)$-nearest neighbor $Z(p)$ over all points in $Z_{i,j}$;

Step 6. For each point p, find the closest of the three points $X(p)$, $Y(p)$, and $Z(p)$, and return it as the $(0, r)$-nearest neighbor of p.

By using the sorting algorithm of [2,3,5], Steps 1 and 2 can be completed in $O(1)$ time on a RMESH of size $n \times n$. In Step 3, a submesh of size $n \times n^{2/3}$ can be assigned to each X_i. Consequently, Step 3 can be completed in $O(1)$ time by the naive algorithm of Lemma 3.2. In the same way, Step 4 can be implemented to run in $O(1)$ time. Step 6 involves only local computation and can be performed, in the obvious way, in $O(1)$ time.

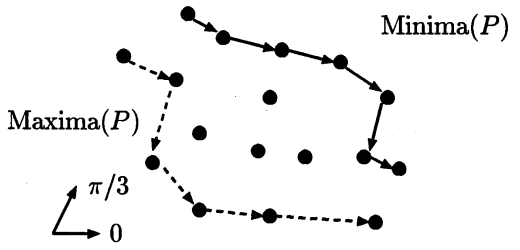

Figure 2: *Illustrating* Maxima(S) *and* Minima(S) *for directions* 0 *and* $\pi/3$

The remainder of this section is devoted to showing that with a careful implementation Step 5 will run in O(1) time. We shall begin by presenting a few technical results that are key in understanding why our implementation works.

Consider, as before, a set $S = \{p_1, p_2, \ldots, p_n\}$ of points. A point p_i in S is a $(0, r)$-*maximal* (resp. *minimal*) point of S if p_i is not $(0, r)$-dominated by (resp. does not $(0, r)$-dominate) any other point in S. We shall use Maxima(S) (resp. Minima(S)) to denote the chain of all maximal points in S specified in counter-clockwise order (resp. all minimal points in clockwise order). These concepts are illustrated in Figure 2 for the directions 0 and $\pi/3$.

Next, we propose to show that only points in Maxima($X_i \cap Y_j$) may have their $(0, r)$-nearest neighbor in $Z_{i,j}$. Moreover, if the $(0, r)$-nearest neighbor of a point in Maxima($X_i \cap Y_j$) lies in $Z_{i,j}$, then it can only be in Minima($Z_{i,j}$).

For later reference, we take note of the following technical lemmas.

Lemma 3.3 *For three points p_1, p_2, p_3 and a direction r $(0 < r < \pi/2)$, if p_1 $(0, r)$-dominates p_2 and if p_2 $(0, r)$-dominates p_3, then $d(p_1, p_2) < d(p_1, p_3)$.* □

Lemma 3.4 ARANN$_r(P, Q)$ *is planar if* $r \leq \pi/2$.

Consider two sets of points P and Q such that all points in P $(0, r)$-dominate all the points in Q.

Let $\text{ARANN}_r(P,Q)$ be a set of edges (p,q) such that p belongs to P, q belongs to Q, and q is the $(0,r)$-nearest neighbor of p.

Lemma 3.5 *If* $(p,q) \in \text{ARANN}_r(P,Q)$, *then* $p \in \text{Minima}(P)$ *and* $q \in \text{Maxima}(Q)$.

Proof. If p does not belong to $\text{Minima}(P)$, then there exists a point p' in $\text{Minima}(P)$ that dominates q. But now, Lemma 3.3 guarantees that $d(p,p') < d(p,q)$, a contradiction. In case q does not belong to $\text{Maxima}(Q)$ we can show a contradiction in an essentially similar fashion. □

Lemma 3.5 has the following important consequence.

Corollary 3.6
$\text{ARANN}_r(\text{Minima}(P), \text{Maxima}(Q)) = \text{ARANN}_r(P,Q)$. □

Corollary 3.6 guarantees that in order to compute $\text{ARANN}_r(P,Q)$ examining all the pairs of points p in P and q in Q is not necessary: all that is needed is to compute $\text{ARANN}_r(\text{Minima}(P), \text{Maxima}(Q))$.

To pursue this idea further, write $\text{Minima}(P) = \{p_1, p_2, \ldots, p_m\}$ and $\text{Maxima}(Q) = \{q_1, q_2, \ldots, q_n\}$ and assume that for some fixed ϵ, $(0 < \epsilon \leq 1)$, $m = n^\epsilon$. Corollary 3.6 motivates the following approach to compute $\text{ARANN}_r(\text{Minima}(P), \text{Maxima}(Q))$.

Let us partition $\text{Minima}(P)$ into $n^{\epsilon/2}$ chains $P_1, P_2, \ldots, P_{n^{\epsilon/2}}$ in such a way that

- $P_1 = \{p_1, p_2, \ldots, p_{n^{\epsilon/2}}\}$ and
- $P_k = \{p_{(k-1)n^{\epsilon/2}+1}, p_{(k-1)n^{\epsilon/2}+1}, \ldots, p_{kn^{\epsilon/2}}\}$, for every k, $2 \leq k \leq n^{\epsilon/2}$.

Refer to Figure 3 for an illustration. For each k $(1 \leq k \leq n^{\epsilon/2})$, let $q_{j_k} (\in Q)$ be the $(0,r)$-nearest neighbor of $p_{kn^{\epsilon/2}}$ over all Q.

Observe that the points q_{j_k} thus defined induce a partition of the set Q into $n^{\epsilon/2}$ chains $Q_1, Q_2, \ldots, Q_{n^{\epsilon/2}}$ such that

- $Q_1 = \{q_1, q_2, \ldots, q_{j_1}\}$ and
- $Q_k = \{q_{j_{k-1}}, q_{j_{k-1}+1}, \ldots, q_{j_k}\}$, for every k, $2 \leq k \leq n^{\epsilon/2}$.

We note that $q_{j_k} \in Q_k \cap Q_{k+1}$. Lemma 3.4 guarantees that in order to compute the $(0,r)$-nearest neighbor of a point p in P_k with respect to $\text{Maxima}(Q)$, we can restrict ourselves to computing the $(0,r)$-nearest neighbor of p over Q_k. In other words, $\text{ARANN}_r(P_k, Q_k) = \text{ARANN}_r(P_k, \text{Maxima}(Q))$. Thus, computing $\text{ARANN}_r(\text{Minima}(P), \text{Maxima}(Q))$ reduces to computing $\text{ARANN}_r(P_k, Q_k)$ for all k.

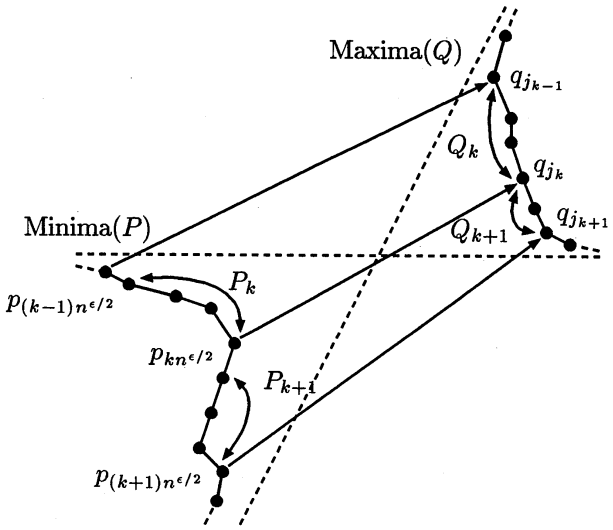

Figure 3: *Partitioning* $\text{Minima}(P)$ *and* $\text{Maxima}(Q)$

The sampling strategy outlined above leads to the following $O(1)$-time algorithm for computing $\text{ARANN}_r(\text{Minima}(P), \text{Maxima}(Q))$ on a RMESH of size $n \times n^\epsilon$, for some fixed $0 < \epsilon \leq 1$. We assume that each point in $\text{Minima}(P)$ has been assigned to one column and each point in $\text{Maxima}(Q)$ has been assigned to one row of the RMESH.

Step 1 Partition columnwise the original RMESH of size $n \times n^\epsilon$ into $n^{\epsilon/2}$ submeshes of size $n \times n^{\epsilon/2}$ each. In the k-th such submesh, $1 \leq k \leq n^{\epsilon/2}$, compute $d(p_{kn^{\epsilon/2}}, q)$ for all points q in $\text{Maxima}(Q)$;

Step 2 For every k, $(1 \leq k \leq n^{\epsilon/2})$, use the k-th submesh to compute

- the $(0,r)$-nearest neighbor q_{j_k} of $p_{kn^{\epsilon/2}}$ by finding the smallest of the distances $d(p_{kn^{\epsilon/2}}, q)$ computed above;
- using the point q_{j_k} determine Q_k.

If $|Q_k| = 1$, then the $(0,r)$-nearest neighbor of every point p in P_k is precisely q_{j_k}, and thus $(p, q_{j_k}) \in \text{ARANN}_r(P_k, \text{Maxima}(Q))$. We

shall, therefore, assume that $|Q_k| \geq 2$ for all k;

Step 3 Partition the given RMESH of size $n \times n^\epsilon$ rowwise into $n^{\epsilon/2}$ submeshes as follows. For each k, $(1 \leq k \leq n^{\epsilon/2})$, the k-th submesh has size $(|Q_k| - 1) \times n^\epsilon$ and involves rows $j_{k-1} - 1$ through j_k of the original mesh. In the k-th submesh compute $\text{ARANN}_r(P_k, Q_k)$ as follows:

Step 3.1 Partition the k-th submesh of size $(|Q_k| - 1) \times n^\epsilon$ into $n^{\epsilon/2}$ submeshes each of size $(|Q_k| - 1) \times n^{\epsilon/2}$. Assign each point p in P_k to each such submesh, and compute $d(p, q)$ for each point q in Q_k.

Step 3.2
By using the algorithm of Lemma 3.1, compute the minimum of all $d(p, q)$ over all q in Q_k in each submesh assigned to p and return the $(0, r)$-nearest neighbor of p.

The reader should have no difficulty to confirm that Steps 1 and 2 can be performed in constant time from Lemma 3.1. By using the prefix-sums algorithm of [6], the partitioning in Step 3 can be performed in constant time. Steps 3.1 and 3.2 can also be implemented to run in constant time. To summarize, we have proved the following result.

Lemma 3.7 *If* Minima(P) *and* Maxima(Q) *have been assigned to the columns and the rows, respectively, of a RMESH of size* $n \times n^\epsilon$, *then* $\text{ARANN}_r(\text{Minima}(P), \text{Maxima}(Q))$ *can be computed in* $O(1)$ *time for every fixed* $0 < \epsilon \leq 1$. □

We now discuss an O(1) time implementation of Step 5 that computes $Z(p)$ for all p. First, we assume a RMESH with $n \times 2n$ processors.

Step 5.1 Partition the $n \times 2n$ RMESH columnwise into $n^{2/3}$ submeshes. For i, j, $(1 \leq i, j \leq n^{1/3} - 1)$ let submesh $\text{R}(i, j)$ be of size $n \times (|X_i \cap Y_j| + n^{1/3})$;

Step 5.2 Compute $\text{Minima}(X_i \cap Y_j)$ in each $\text{R}(i, j)$;

Step 5.3 Compute $\text{Maxima}(Z_{i,j})$ in each $\text{R}(i, j)$;

Step 5.4
Compute the graph $\text{ARANN}_r(\text{Minima}(X_i \cap Y_j), \text{Maxima}(Z_{i,j}))$ in each $\text{R}(i, j)$. If $(p, q) \in \text{ARANN}_r(\text{Minima}(X_i \cap Y_j), \text{Maxima}(Z_{i,j}))$ then return $Z(p) = q$.

Step 5.1 is complicated, because the number of columns of each submesh is different. The partitioning specified in Step 5.1 can be completed in $O(1)$ time by using the sorting algorithm of [2, 3, 5]: sort the n points in lexicographical order of $(x'(p), y(p))$. Clearly, for each i and j, all points in $X_i \cap Y_j$ are consecutive in the sorted points. If the smallest point in $X_i \cap Y_j$ and the largest one are s-th and t-th in the sorted order, then the submesh $\text{R}(i, j)$ is assigned columns $s + (i \cdot n^{1/3} + j) \cdot n^{1/3}$ to $t + (i \cdot n^{1/3} + j + 1) \cdot n^{1/3}$. Hence, Step 5.1 can be completed in $O(1)$ time. Steps 5.2 can be completed as follows: Let $X_i \cap Y_i = \{p_1, p_2, p_3, \ldots\}$, where $x'(p_k) \leq x'(p_{k+1})$ for all k. Compute the suffix-maxima of $\{y(p_1), y(p_2), y(p_3), \ldots\}$, by the algorithm of Lemma 3.1. Then, $p_k \in \text{Minima}(X_i \cap Y_j)$ if and only if $\max\{y(p_{k+1}), y(p_{k+2}), y(p_{k+3}), \ldots\} > y(p_k)$. Therefore, $\text{Minima}(X_i \cap Y_j)$ can be computed in $O(1)$ time. Step 5.3 can be completed in the same way. To apply the algorithm of Lemma 3.7 to Step 5.4, a serial number must be assigned to the points in $\text{Minima}(X_i \cap Y_j)$ and those in $\text{Minima}(Z_{i,j})$. These numbering can be obtained in the obvious way by using the prefix-sum algorithm of [6]. Then, by executing the algorithm of Lemma 3.7, Step 5.4, can be completed in $O(1)$ time. Therefore, the ARANN problem can be solved in $O(1)$ time on an $n \times 2n$ RMESH.

Since the algorithm above that uses $n \times 2n$ processors, can be implemented on an $n \times n$ RMESH by a simple scheduling technique, the ARANN problem can also be solved in $O(1)$ time on an $n \times n$ RMESH. Furthermore, the ARANN problem for the obtuse angle can be computed easily by partitioning the angle into several acute angles, solving the ARANN for each angle. Thus, we have proved the following result.

Theorem 3.8 *Given an arbitrary set of* n *points in the plane and a direction* r, $(0 < r \leq 2\pi)$, *the corresponding instance of the ARANN problem can be solved in* $O(1)$ *time on a reconfigurable mesh of size* $n \times n$. □

References

[1] J. Jang and V. K. Prasanna, Parallel geometric

problems on the reconfigurable mesh, *Proc. of the International Conference of Parallel Processing,* St. Charles, Illinois, III, 1992, 127–130.

[2] J. Jang and V. K. Prasanna, An optimal sorting algorithm on reconfigurable meshes, *Proc. International Parallel Processing Symposium*, 1992, 130–137.

[3] R. Lin, S. Olariu, J. L. Schwing, and J. Zhang, Sorting in O(1) time on a reconfigurable mesh of size $N \times N$, *Parallel Computing: From Theory to Sound Practice, Proceedings of EWPC'92*, Plenary Address, IOS Press, Amsterdam, 1992, 16–27.

[4] R. Miller, V. K. P. Kumar, D. Reisis, and Q. F. Stout, Parallel Computations on Reconfigurable Meshes, *IEEE Transactions on Computers*, 42, (1993), 678–692.

[5] M. Nigam and S. Sahni, Sorting n numbers on $n \times n$ reconfigurable mesh with buses, *Proc. 7th International Parallel Processing Symposium*, April 1993, 174–181.

[6] S. Olariu, J. L. Schwing, and J. Zhang, Fundamental Algorithms on Reconfigurable Meshes, *Proc. 29–th Annual Allerton Conf. on Communication, Control, and Computing*, 1991, 811–820.

[7] F. P. Preparata and M. I. Shamos, *Computational Geometry – An Introduction*, Springer-Verlag, Berlin, 1990.