CHAPTER II

LITERATURE SURVEY

2.1 Introduction

In relevance to our problems in question, we made a literature survey that throws light on what have been accomplished until our contribution and the survey findings in the respective domains of the problems are presented in separate sections of this chapter. The complete descriptions followed by solutions presented in the form of pseudo-code for the problems are dealt individually one after the other in ensuing chapters.

In each section, we introduce a problem or a class of problems with its definition in English sentences and the survey results in the respective domain. Finally, the limitations of the literature, which turned as the motivation for our work, are summarized.

2.2 The LCHP Problem

The LCHP problem is to locate a length-constrained heaviest path (LCHP) on a tree, given weight and length on each edge of the tree. It finds application in sub-fields such as Network Design. For instance, given a tree network with length and weight on each edge, it is desired to upgrade the network by replacing a path with high-speed edges. When the length of an edge represents the building cost and the weight represents the profit such as traffic load, our objective is to find a LCHP given a budget constraint that limits the length of the path to be upgraded, with maximum profit.
There is another similar kind of problem found in the literature. It is to find an optimal position with relative distances in an evolutionary tree. Constructing an evolutionary tree (phylogenetic tree) according to pairwise dissimilarities has been a useful model for analyzing evolutionary relations between species. An evolutionary tree is a rooted tree with leaves as species and weights on edges. The distance between two species is the total weight of the path between the two leaves. Wu and Tang [22] gave a linear time solution to the problem.

Wu et al. [23] presented an efficient sequential algorithm for the LCHP problem. Their algorithm employs divide-and-conquer strategy and runs in $O(n \log^2 n)$ time while brute-force algorithm takes $O(n^2)$ time, where $n$ is the number of vertices of the input tree. In addition, they established that the same algorithm requires only $O(n \log n)$ time when the edge lengths are all integers ranging from 1 to $O(n)$.

The underlying technique of the algorithm is as follows: first, it roots the tree at centroid (a vertex, which when deleted along with its edges incident on, leaves the tree with all subtrees each having no more than $n/2$ vertices), determines the best path containing the centroid, and then finds the best path within each subtree by recursively calling the algorithm. Such a technique had been used in some other path problems also. For instance, the technique was used to find the $k^{th}$ longest path in a tree [24]. When many known algorithms were based on selection in a set, whose cardinality was superlinear in terms of the input length, Megiddo et al. [24] aspired, in these cases to have selection algorithms that could run in sublinear time in terms of
the cardinality of the set. The methods developed can be applied to improve
the previously known upper bounds for the time complexity of various
location problems. Peng and Lo [25] also employed centroid-based technique
to find the core of a tree with specified length and designed efficient
algorithms for both sequential and parallel computational models. The
algorithms can be readily extended to a tree network in which arcs have
nonnegative integer lengths. The results presented might provide a basis for
the study of other facility shapes such as trees and forest of fixed sizes.

Recently, Becker et al. [26] also have employed the same strategy in finding
the $q$-core of a tree. A $q$-core of an edge-weighted tree $T$ is a set of $q$
mutually disjoint paths in $T$ that minimizes the sum of the distances of all
vertices in $T$ from any of the $q$ paths, where $q \geq 1$ is an integer. Peng and Lo
[25] had claimed that their method could be applied to trees with integer
lengths, but no details are given and the extension is not immediately
obvious. But, Becker et al. [26] entertained an extension of the latter one.
They also considered the $q$-core problem both in unweighted trees and in
weighted ones. Then, two algorithms were provided, one with time
complexity of $O(nq)$ for unweighted trees and the other for weighted trees
which runs in $O(n \cdot \log^2 n)$ time. Recently, Wang [27] has proposed a linear
time algorithm for finding 2-core of a tree. It was also claimed that the
proposed algorithm can be implemented on the EREW PRAM in $O(\log^2 n)$
time using $O(n \log n)$ cost, with some modifications.

To the best of our knowledge, there is no parallel solution to the
problem in question prior to our contribution [9]. Motivated thus, we have
solved the problem on CREW PRAM model and also established that the
problem can easily be solved on a weaker model, EREW PRAM model, with the same cost.

It is worthy to mention that several other similar problems can also be solved with the same cost by appropriately modifying our proposed parallel algorithm. As, essentially, we have provided a parallel solution for a tree with edges assigned two real numbers, length and weight, we can easily tailor it up to solve the following four problems: Given a tree with edge length and edge weight, find a maximum (minimum) weighted path on the tree on condition that the edge length is no more than (or less than) a given value, a budget constraint.

There are many parallel solutions for problems on CREW PRAM model found in the literature. We mention a few: Shiloach and Vishkin [28] employed the model to solve the problems of finding the maximum, merging, and sorting by N processors. The main results are: 1) Finding the maximum of n elements (1 < N ≤ n) within a depth of O(n/N + log log N); (optimal for N ≤ n/log log n). 2) Merging two sorted lists of length n1 and n (n1 ≤ n) within a depth of O(n/N + log n) for N ≤ n (optimal for N ≤ n/log log n). 3) Sorting n elements within a depth of O((n/N) log n + log n log N) for N ≤ n, (optimal for N ≤ n / log n). Cole [29] gave a parallel implementation of merge sort on a CREW PRAM that requires n processors and O(log n) time; the constant in the running time is small. He also imparted a more complex version of the algorithm for the EREW PRAM; it also requires n processors and O(log n) time. The constant in the running time is still moderate, though not as small. His work has many citations in the literature.
Aggarwal and Park employed the model to search in parallel in multidimensional monotone arrays [30]. A two-dimensional matrix $M'$ is called monotone if the maximum entry in its $i^{th}$ row lies below or to the right of the maximum entry in its $(i-1)^{th}$ row. A Matrix is called totally monotone if every $2 \times 2$ sub-array is monotone. It had been shown in the literature already that several problems in computational geometry could be reduced to the problem of finding row maxima in totally monotone matrices. They generalized the notion of two-dimensional monotone matrices to multidimensional matrices and exhibited a wide variety of problems involving computational geometry, dynamic programming, VLSI river routing, and finding certain kinds of shortest paths that could be solved efficiently by finding maxima in totally monotone matrices.

Reif and Sen gave a new randomized sampling technique in Computational Geometry [31] on the same model. Their technique called Polling has applications to deriving efficient parallel algorithms. As an example of its use in computational geometry, they presented an optimal parallel randomized algorithm for intersection of half-spaces in three dimensions. Because of well-known reductions, their methods also yield equally efficient algorithms for fundamental problems like the convex hull in three dimensions, Voronoi diagram of point sites on a plane and Euclidean minimal spanning tree. Their algorithms run in time $t(n) = O(\log n)$ for worst-case inputs and use $p(n) = O(n)$ processors in a CREW PRAM model where $n$ is the input size. They are randomized in the sense that they use a total of only $O(\log^2 n)$ random bits and terminate in the claimed time bound with
probability $1 - n^x$ for any $e > 0$. They are also cost-optimal since the sequential time bound for all these problems is $\Omega(n \cdot \log n)$. The best known deterministic parallel algorithms for 2-d Voronoi-diagram and 3-d Convex hull, prior to their work, were with $O(\log^n n)$ and $O(\log^2 n \cdot \log \log n)$ time respectively.

Boxer [32] used the model for finding congruent regions in parallel. Given a straight-line embedded plane graph $G$ of $e_1$ edges and a polygon $P$ of $e_2$ edges, $e_2 \leq e_1$, he described an algorithm for finding all polygons in $G$ that are congruent to $P$. His algorithm requires $\Theta(e_1 \cdot \log e_1)$ time for a CREW PRAM with $e_2$ processors. He also showed that the problem is in NC by showing how to implement our algorithm in $\Theta(\log e_1)$ time using $e_1 \cdot e_2$ processors.

Very recently, Lin and Chung have found a space-efficient Huffman Decoding algorithm on CREW PRAM [33]. They have first transformed the Huffman tree into a single-side growing Huffman tree, then presented a memory-efficient data structure to represent the single-side growing Huffman tree, which requires $(n + d') \cdot \lceil \log n \rceil$-bits memory space, where $n$ is the number of source symbols and $d'$ is the depth of the Huffman tree. Based on the proposed data structure, they developed an $O(d')$-time Huffman decoding algorithm. They finally modified their proposed data structure to design an $O(1)$-time parallel Huffman decoding algorithm on CREW PRAM using $d'$ processors.

On CREW PRAM, Xue [34] developed a cost-optimal parallel algorithm for computing force field in N-body simulations. He considered the
following force field computation problem: given a cluster of \( n \) particles in three-dimensional space, compute the force exerted on each particle by the other particles. Depending on different applications, the pairwise interaction could be either gravitational or Lennard-Jones'. In both cases, the force between two particles vanishes as the distance between them approaches to infinity. Since there are \( n(n-1)/2 \) pairs, direct method requires \( \Theta(n^2) \) time for force evaluation, which is very expensive for astronomical simulations. In [34], he presented an algorithm which computes the force field in \( \Theta(\log n) \) time on a \( \Theta(n / \log n) \) processor CREW PRAM and hence cost-optimal. His parallel algorithm also yields a new \( \Theta(n) \)-time sequential algorithm for force field computation.

### 2.3 Some Basic Geometric Problems on LARPBS

Attracted by the practicality of the LARPBS model over PRAM models [16], we tried some basic geometric problems, viz., leftmost-one computation, maximal elements problem and range minima related problems, with a view to designing cost optimal parallel algorithms for. Many parallel solutions have been proposed on the LARPBS model in the literature in the recent years. The inventors of the model, Pan and Li, themselves, have solved many problems employing the LARPBS model [35-42].

Pan et al. [35] implemented a parallel quicksort algorithm on the model, and analyzed its time complexity. For a set of \( N \) numbers, the quicksort algorithm reported in [35] runs in \( \mathcal{O}(\log N) \) average time on LARPBS of size \( N \). If the number of processors available is reduced to \( N' \), where \( N' < N \), the algorithm runs in \( \mathcal{O}((N/N') \log N) \) average time and is still
scalable. Besides proposing a new algorithm on the model, some basic data movement operations involved in the algorithm were also discussed. Li et al. [36] presented efficient parallel matrix multiplication algorithms for LARPBS. By exploiting the reconfigurability of LARPBS, they established that the model is able to support parallel implementations of divide-and-conquer computations like Strassen's algorithm. The main contributions of [36] are five matrix multiplication algorithms with varying degrees of parallelism on the LARPBS. Again, Li et al. [37] exhibited efficient deterministic and probabilistic methods for simulating PRAM computations on LARPBS. Many results were established with a view to proving that an LARPBS can simulate a PRAM very efficiently. Li and Pan improved their matrix multiplication algorithms by parallelizing the fastest sequential matrix multiplication algorithm on LARPBS in [38]. The algorithm has linear speedup and cost-optimality in a wide range of choice for the number of processors. It turns out that, for parallel matrix multiplication, a distributed memory system with optical interconnections like LARPBS compares favorably with shared memory systems, where the concurrent read capability is replaced by highly efficient communications with predictable senders and receivers. In the field of Image Processing too, Pan and Li have been contributing many findings. For instance, Pan et al. [39] examined the possibility of implementing the problem of finding a distance map for an image efficiently using an optical bus. Computing a distance map (distance transform) is an operation that converts a two-dimensional image consisting of black and white pixels to an image where each pixel has a value or a pair of coordinates that represents the distance to or location of the nearest black
pixel. It is a basic operation in image processing and computer vision fields, and is used for expanding, shrinking, thinning, segmentation, clustering, computing shape, object reconstruction, etc. They showed that the problem for an \( n \times n \) image can be implemented in \( O(\log n, \log \log n) \) bus cycles deterministically or in \( O(\log n) \) bus cycles with high probability on an LARPBS with \( n^2 \) processors. For processor arrays with practical sizes, a bus cycle is roughly the time of an arithmetic operation. Hence, the algorithm compares favorably to the best known parallel algorithms for the same problem in the literature. Han et al. [40] showed that the selection problem can be solved on the LARPBS model in \( O((\log \log N)^2/ \log \log \log N) \) bus cycles deterministically. Similarly, related to Image Processing again, Chen et al. [41] proposed efficient Euclidean Distance Transform algorithms. Very recently, Bourgeois et al. [42] have developed constant time fault tolerant algorithms on LARPBS. The only previous work in this area considers faulty processors or hardware for an LARPBS, resulting in fault tolerant algorithms that run in \( O(\log N) \) time. They have extended and improved upon previous results by considering new fault scenarios and slight modifications to the LARPBS model. By modifying the switches on the receiving segment of an LARPBS, a reasonable, low-cost extension to the model, they are able to achieve fault tolerant algorithms that execute in \( O(1) \) time rather than \( O(\log N) \) time. The fault tolerant algorithms that they considered are key basic fundamental algorithms, such as compaction, binary prefix sums, and sorting, that all execute in constant time on a healthy LARPBS.

Datta has contributed few solutions on the model [43, 44, 46]. Datta et al. [43] presented two fast sorting algorithms on LARPBS, first one with
$O(\log N \cdot \log \log N)$ worst case time using $N$ processors and the second one with $O((\log \log N)^2)$ worst case time using $N^{1+z}$ processors, for any fixed $z$ such that $0 < z < 1$. Their algorithms are based on a novel deterministic sampling scheme for merging two sorter arrays of length $N$. Datta individually solved graph-theoretic problems on LARPBS [44]. Prior to his work, Li et al. [45] had solved a number of important and interesting problems from graph theory on an LARPBS. Their algorithms are based on their own fast matrix multiplication and extreme value (maximum/minimum) finding algorithms. But, Datta proposed more efficient algorithms for solving several fundamental graph-theoretic problems on LARPBS. His algorithms include finding connected components, minimum spanning forest, biconnected components, bridges and articulation points for an undirected graph. One can compute the connected components and minimum spanning forest of a graph in $O(\log |V|)$ time using $O(|V| + |E|)$ processors where $V$ and $E$ denote the sets of vertices and edges of an undirected graph respectively, with his algorithms. Recall that $|E| = O(|V|^2)$ for a dense graph. Both the processor and time complexities of these two algorithms match the complexities of algorithms on the Arbitrary and Priority CRCW PRAM models which are two of the strongest PRAM models. Very recently, Datta et al. [46] have contributed an efficient algorithm for Euclidean distance transform by employing their own novel deterministic sampling scheme.

We also have designed and analyzed four algorithms on the LARPBS model. Of them, one is on Huffman Decoding. Kannan et al. [47] have designed a parallel algorithm for Huffman Decoding with constant time complexity employing Lin and Chung's memory-efficient data structure [33]
for Huffman tree. And, the rest three related to Computational Geometry, are dealt individually each in a chapter shortly.

Now, we draw the attention to some basic geometric problems and their respective solutions found in the literature. In many applications related to Pattern Recognition, Computer graphics and Robotics, geometric problems involving basic questions on sets of points, polygons etc., are encountered and it is realized to have algorithms with speeds close to real time.

Especially, in computational areas such as Image Processing, Digitized Geometry and Computer Graphics, there are requirements to solve leftmost-one problem felt very often. The problem is to determine the position of the leftmost unity (one), if any, in each row of a matrix. Stout [48] proposed an $O(n^{1/m})$ time taking parallel algorithm to the leftmost one problem on a mesh with row buses. Gurla [49] presented an $O(n^{1/6})$ time parallel algorithm on a $\sqrt{n} \times \sqrt{n}$ mesh with row broadcasting, each processor having a fixed number of registers. However, it is required to design a constant time solution for the problem, as it is generally part of another advance problem and the time requirement of the leftmost one problem or any other similar basic problem is to be restricted to as minimum as possible in order to enable finding efficient solution for the advance problem. Here, it is desired to specify the significance of the LARPBS model. Compared to PRAM models, LARPBS is stronger in the aspect that it is possible to have a constant-time taking parallel algorithm on an LARPBS model whereas the most realistic model, EREW PRAM model, can’t let the users design algorithms with less than $O(\log N)$ time as broadcasting of a datum to various
processors of the model itself requires $O(\log N)$ time. Hence, we have proposed an $O(1)$-time simple parallel algorithm for the leftmost-one problem with an LARPBS model. Later, Chung and Lin [50] employed an enhanced two-dimensional mesh with segmented row buses (2-MSRB) to develop a parallel algorithm for solving the leftmost-one problem. Given an image of size $\sqrt{n} \times \sqrt{n}$ with entry being a 0 or a 1, their algorithm runs in $O(n^{1/8})$-time on a $\sqrt{n} \times \sqrt{n}$ 2-MSRB.

The second basic problem, the maximal elements problem is defined as follows: Between two points of Cartesian plane, we bring in a binary relation, informally known as domination. One point is said to dominate other when both of its co-ordinates have greater values than those corresponding of the other. We designate a point as maximal when it has no dominators. We intend to find all maximal elements of $P$, called $m$-contour $m(P)$.

Dehne [51] had solved this problem with an $O(\sqrt{n})$-time parallel algorithm on a mesh connected computer. In [51], considering a set $S$ of $n$ points in the Euclidean plane, we find efficient parallel algorithms proposed for the following two problems: (i) compute the contour spanned by the maximal elements of $S$, and (ii) compute the number of dominated points for every element of $S$. Both algorithms run in $O(\sqrt{n})$-time on a mesh of $n$ processors, which is asymptotically optimal since any nontrivial computation requires $\Omega(\sqrt{n})$ time on the mesh. The algorithms can be generalized to solve the $d$-dimensional maximal elements and searching problem in $O(n^{1/2 + \log(d - 2)})$ time.
In [52], Atallah and Goodrich solved the 2-d maximal elements problem in $O(\log n)$ time using an $O(n \log n)$ sorting network [53] and employing parallel prefix [54]. They gave general techniques for solving a number of geometric problems whose efficient sequential algorithms use the plane-sweep paradigm. These techniques can be viewed as efficient parallel analogues to the plane-sweeping paradigm. They applied the plane-sweep tree technique to intersection detection, trapezoidal decomposition, polygon triangulation, maximal elements finding and planar point location.

Stojmenovic and Miyakawa [55] developed an $O(\log n)$ time optimal algorithm for solving the maximal elements problem in the plane using $O(n)$-processor CREW PRAM model. Recently, Thangavel [56] has proposed an $O(1)$-time algorithm employing $O(n^2)$ processors with a two dimensional processor arrays with reconfigurable bus system. As the cost is $O(n^3)$, we got motivated to design an $O(\log n)$-time parallel algorithm using an $O(n)$-processor LARPBS model, which is cost optimal too.

Finally, a class of range minima and range co-minima and nearest neighbor problems has been dealt. The range minima problem is: Given a set of real elements $S = \{s_1, s_2, ..., s_n\}$, and two query indices $1 < i < j < n$, compute $\min(s_i, s_{i+1}, ..., s_j)$. The range co-minima problem is: Given an integer set $S = \{s_1, s_2, ..., s_n\}$ with $1 < s_i < n$, and two query indices $1 < i < j < n$, find the least positive integer that is not included in $\{s_i, s_{i+1}, ..., s_j\}$.

Range minima problem has an efficient sequential solution by Kim [57]. By addressing a geometric problem called the segment dragging problem, he solved the range minima. The segment dragging problem is
described as follows: We have \( n \) "obstacles" in the plane and want to preprocess them so that, given a query vertical line segment \( s \), intersecting no obstacles, the first obstacle hit by \( s \) when we drag \( s \) horizontally to the right can be found efficiently. He presented an \( O(\log n) \) time, \( O(n) \) processor parallel algorithm for preprocessing when the obstacles are points or nonintersecting line segments. After preprocessing, a query can be answered in \( O(\log n) \) time using a single processor. He had employed the EREW PRAM model for solving the problem.

Recently, Kim [58] has proposed a parallel algorithm for range minima and co-minima employing a Reconfigurable Mesh (RMESH). His preprocessing sub-algorithms for both the problems take constant time on an \( n \times n \) RMESH and a query can be answered in constant time using a single processor. Tsai et al. [59] employed a hyper-bus broadcast network (HBBN) which consists of processors only sharing by some global buses, and there are no local links between processors. Based on such architecture, they proposed several efficient time parallel algorithms for solving the well-known fundamental data movement problems which include the leftmost one problem, the prefix maxima/minima problem, the \( m \)-contour problem, the all nearest neighbor problem and the all nearest smaller values problem, respectively.

We got motivation from Kim's work that with only \( n \) processors a parallel algorithm be achieved. For, Kim observed that reconfigurability and the data communication ability through sub-buses in constant time enabled the achievement. Even on CRCW PRAM models, the most powerful among
the SM SIMD variants, it is not guaranteed that these problems can be solved in constant time using only $O(n^3)$ processors, as those variants were used to develop efficient parallel algorithms without considering communications among processors. The actual hindrance in designing parallel algorithms on CRCW PRAM models is that there is no easy way to group processors so that those in the same group can access a value in constant time.

Here, we wish to recall the observation made by inventors of LARPBS, Pan and Li, in [16] which is very close to Kim’s observation. These ideas turned our focus of research toward employing LARPBS for designing parallel algorithms.

We intend to solve two other related problems called nearest lower neighbor and nearest upper neighbor problems, also. Nearest lower neighbor problem, given a set $S = \{s_1, s_2, \ldots, s_i, s_{i+1}, \ldots, s_j, \ldots, s_n\}$ and query indices $i$ and $j$, is to find $s_a$ in $S$ such that $s_a < s_k$ for $k = i, i+1, \ldots, j$ and there is no other $s_b > s_a$ which satisfies this condition. One can easily define nearest upper neighbor problem from the above definition. Thangavel [56] have solved the problem on an $n \times n$ RMESH. The solutions take constant time and the basic strategy employed is constant time sorting on an $n \times n$ RMESH. Driven by the need for constant time parallel algorithms with only $O(n)$ processors, we also have attempted to solve these four aforementioned problems and succeeded with $O(1)$ time parallel algorithms.

2.4 An Advance Geometric Problem – Rectilinear 3-centers

The set of solutions proposed for the class of basic geometric problems may make one develop an opinion that the LARPBS model with its
constant time taking primitive functions (almost all) is suitable for fundamental problems only. But, one can find in the literature that many advance problems have been solved on the LARPBS model, matrix multiplication [36, 38] being one among them. Furthermore, LARPBS model is appropriate for communication-oriented problems rather than computation-oriented, as the constant-time primitive operations of the model are communication-oriented [16]. Despite Computational Geometric problems are computation-oriented, we have intended to solve an advance computational geometric problem, the rectilinear $q$-center problem. It is to determine the centers of at most $q$ congruent axis-parallel squares of minimal size whose union covers a given finite set $P$ of points, $P \subseteq \mathbb{R}^2$. The problem has its application in facility location. When there is a need to locate $q$ supply points for a given set of demand points such that for any demand point the nearest supply is as close as possible. On a geometric platform, we represent the supply and demand points as points in $\mathbb{R}^d$. The metric followed to measure the distance between points is rectilinear metric ($l_1$ or $l_\infty$). One has to be warned that solving for any $q$ leads to an NP-complete problem. Therefore, we confined $q$ to be equal to 3.

According to Tamir [60], the most common problems studied in network location theory are the $q$-center and the $q$-median problems. In the $q$-center problem the objective is to locate $q$ service facilities to minimize the maximum of the service distances of the $n$ customers to their respective nearest service facility, and in the $q$-median model the objective is to minimize the sum of these $n$ service distances (A customer is served only by the closest facility). He studied the multi-facility $q$-centrum model that
generalizes and unifies the above problems. The objective of this unifying model was to minimize the sum of the \( q \) largest service distances. The \( q \)-center and the \( q \)-median problems correspond to the cases where \( q = 1 \) and \( n \), respectively. He presented polynomial time algorithms for solving the multi-facility \( q \)-centrum problem on path and tree graphs. His algorithms are centroid based. There is a mention of linear time algorithm for finding centroid of a tree.

For \( q \leq 3 \), Sharir and Welzl [61] gave a solution with linear time. They considered the \( q \)-piercing problem (given a collection of regions), which is to determine whether there exists a set of \( q \) points that intersects each of the given regions. They gave linear or near-linear algorithms for small values of \( q \) in cases where the given regions are either axis-parallel rectangles or convex c-oriented polygons in the plane (i.e., convex polygons with sides from a fixed finite set of directions). They also investigated the planar rectilinear (and polygonal) \( q \)-center problem, in which one is given a set \( S \) of \( n \) points in the plane, and wish to find \( q \) axis-parallel congruent squares (isothetic copies of some given convex polygon, respectively) of smallest possible size whose union covers \( S \). They also studied several generalizations of these problems. New results are a linear-time solution for the rectilinear 3-center problem (by showing that this problem can be formulated as an LP-type problem and by exhibiting a relation to Helly numbers). They presented \( O(n \log n) \)-time solutions for 4-piercing of translates of a square, as well as for the rectilinear 4-center problem; this is worst-case optimal. They devised \( O(n \text{ poly}(\log n)) \)-time solutions for 4- and 5-piercing of axis-parallel rectangles, for more general rectilinear 4-center problems, and for rectilinear 5-center problems.
The 2-pierceability of a set of \( n \) convex \( q \)-oriented polygons can be decided in time \( O(q^2 n \log n) \), and the 2-center problem for a convex \( q \)-gon can be solved in \( O(q^3 n \log n) \) time. The first solution is worst-case optimal when \( c \) is fixed.

Specifically, first, Drezner [62] proposed a linear time algorithm for \( q = 2 \). An \( O(n) \) algorithm for the 1-center problem, an \( O(n) \) algorithm for the 2-center problem, and an \( O(n \log n) \) algorithm for the 3-center problem were given. He also discussed how to generalize these algorithms to general \( q \)-center problems. Then, Nussbaum [63] and Segal [64] presented optimal solutions for the same problem with \( q = 4 \) and \( q = 5 \) respectively.

Nussbaum [63] presented several algorithms for solving rectilinear \( q \)-piercing problem in the plane and in the space. The main results are optimal solution \( O(n \log n) \) for the general rectilinear 4-piercing problem (not only for translate of squares). He also presented an optimal \( O(n \log n) \) solution to the 5-piercing problem and the 5-center problem. His results imply that for \( q \geq 5 \) we can solve the problem in \( O(n^{9/4} + n \log n) \) and raised the question whether a lower bound can be found for the problem. He also observed that these results improved the weighted rectilinear 4-center and 5-center (the squares can be of different sizes) problems. Similarly, Segal [64] also tackled the problem for 5-center problem through the study on piercing of axis-parallel rectangles and rings.

Agarwal and Procopiuc [65] developed a generalized algorithm for any \( q \), which is asymptotically the best algorithm known so far. They provided a general method for solving a large class of clustering problems in sub-exponential time in any fixed dimension. Their method can be adapted to
handle some additional restrictions on the clustering (e.g., discrete \(q\)-center), and works for any metric satisfying a small set of properties. They also observed that the approximation algorithm described by them could be further improved by choosing a non-uniform grid.

Many variants for the \(q\)-center problem have also been studied:

First, we concentrate on the discrete \(q\)-center problem. Agarwal et al. [66] presented an \(O(n^{d/3} \cdot \log^5 n)\)-time algorithm for the planar discrete 2-center problem. They observed that the running time can be improved by a logarithmic factor by exploiting the special structures of canonical subsets and using fractional cascading. It, however, remains a challenging open problem whether there exists a near-linear algorithm for this problem.

Secondly, we consider another variant: the weighted \(q\)-center problem. Ko and Ching [67] solved the weighted tailored 2-partition problem and the weighted 2-center problem under \(l_\infty\)-distance. They designed an \(O(2^{d-1} \cdot d \cdot n)\) algorithm to solve the weighted tailored 2-partition problem and an \(O(d^2 n + d^2 \cdot \log^6 d)\) time algorithm to solve the weighted 2-center problem in the \(d\)-dimensional case.

Thirdly, we discuss a variant: the capacitated \(q\)-center problem. Khuller and Sussmann [68] handled the capacitated \(q\)-center problem which is a modified facility location problem, where one is expected to locate \(q\) facilities in a graph and to assign vertices to facilities, so as to minimize the maximum distance from a vertex to the facility to which it is assigned. Moreover, each facility may be assigned at most \(n_1\) vertices. This problem is
known to be NP-hard. They gave polynomial time approximation algorithms for two different versions of this problem that achieve approximation factors of 5 and 6. They also studied some generalizations of this problem.

Finally, we discuss yet another variant: the Euclidean $q$-center problem. Meggido and Supowit [69] observed that, given $n$ demand points in the plane, the $q$-center problem is to find $q$ supply points (anywhere in the plane) so as to minimize the maximum distance from a demand point to its respective nearest supply point. The $q$-median problem is to minimize the sum of distances from demand points to their respective nearest supply points. They proved that the $q$-center and the $q$-median problems relative to both the Euclidean and the rectilinear metrics are NP-hard. In fact, they proved that it is NP-hard even to approximate the $q$-center problems sufficiently closely. The reductions are from 3-satisfiability.

Recently, Hoffmann [70] has solved the problem for $q = 3$ in linear time. It is simple in the sense that it doesn’t employ any sophisticated data structures or strategies. It employs only median finding. There is no parallel counterpart for the Hoffmann’s sequential algorithm in the literature and we have proposed a parallel solution on the LARPBS model. The development goes as follows: We have, first, developed a constant time parallel selection algorithm for median finding on an LARPBS and employ the same in parallelizing of Hoffmann’s. Previously, Pan [71] solved the selection problem in parallel on LARPBS with an algorithm of $O(n \cdot \log \log n/N)$ time with $N$ processors for an input size of $n$. Rajesäkaran and Sahni [72] developed efficient algorithms for sorting, selection, and packet routing on
the AROB (Array with Reconfigurable Optical Buses) model. One of their sorting algorithms sorts $n$ general keys in $O(1)$ time on an AROB of size $n^\varepsilon \times n$ for any constant $\varepsilon > 0$. They also showed that selection from out of $n$ elements can be done in randomized $O(1)$ time employing $n$ processors.

Our parallel selection solution bases its strategy on a sequential idea similar to Blum et al. [73]. They established that the number of comparisons required to select the $k^{th}$ smallest of $n$ numbers is at most a linear function of $n$ by analysis of a new selection algorithm-PICK. Specifically, no more than $5.4305 \cdot n$ comparisons are ever required. This bound improves for extreme values of $q$. They found a new lower bound on the requisite number of comparisons.

We have proposed a parallel algorithm on an LARPBS based on our own parallel selection algorithm on the same model for the rectilinear 3-center problem and claim the parallel running time to be $O(\log \log n)$ with $O(n)$ processors and hence to be nearly cost optimal, as the lower bound is $\Omega(n)$ [61]. We have also designed and analyzed the performance of the same parallel algorithms on two recently proposed models: RLARPBS (the restricted LARPBS) and LARPBS(p) (the parameterized LARPBS) models, the variants of the base model—the LARPBS.

2.5 A Robotics problem

We later turned our focus on an application problem in the field of Robotics and chose to solve the Robot Path Planning problem. It is to determine a collision-free path for a robot from its start state to end state in a
workspace amidst obstacles. There are many approaches to this problem of which one popular technique is through the construction of a Voronoi Diagram (VD) [74]. Traditionally, this approach involves an assumption to treat a robot as a point and suitably to expand obstacles [74, 75]. For instance, Rao et al. [75] considered a problem of learned navigation of a circular robot, of radius \( r \geq 0 \), through a terrain whose model is not a priori known. They entertained two-dimensional finite-sized terrains populated by an unknown (but, finite) number of simple polygonal obstacles. The number and locations of the vertices of each obstacle are unknown to the robot; but it is equipped with a sensor system that detects all vertices and edges that visible from its present location. With this background, they handled two problems: the visit problem and the terrain model acquisition problem.

Here, we wish to mention a practical application [76] for the problem in question: The visit problem and the terrain model acquisition problem are simplified versions of problems that arose in a practical application. The HERMIES-II (hostile-environment robotic-machine-intelligence experiment series) is a mobile robot developed at the Center for Engineering Systems Advance Research, Oak Ridge National laboratory. The robot was intended for autonomous operation in radiation-prone environments that are unsuitable for humans, such as nuclear power plants experiencing radiation leaks. Autonomous navigation is one of the most important aspects to such a rescue robot. Leaks are infrequent, so a dedicated rescue robot would normally stay idle between rescue missions.
In the visit problem, the robot is required to visit a sequence of destination points. In the terrain model acquisition problem the robot is required to acquire the complete information on the model of the terrain. They presented an algorithmic framework, based on a retraction of free-space onto the Voronoi diagram, to solve both the navigational problems mentioned above. They developed a geometric structure, called the navigation course, based on the retraction of the terrain. The robot employs the navigation course as the roadmap in solving both the problems. The significance of the framework presented is that any graph structure that satisfies the properties of finiteness, connectivity, local constructability, and terrain visibility can be used to solve the problems. They observed that such navigation courses, based on more general retractions, can be obtained in cases such as three-dimensional terrains, polygonal robots, robots composed of linkages and the like.

But, we adhere to another approach that obtains a binary image of the complete workspace and retains the robot as is (No reduction to a single point is assumed here) [77]. As binary image is employed, the discrete VD is needed in lieu of its classical counterpart. The other compelling reasons for employing discrete VD have been discussed in detail in [78].

Arcelli and Baja [78] adopted the 4-metric to construct the Voronoi diagram of a binary digital picture, whose foreground consists of arbitrarily shaped components. The 4-distance transform of the background was employed to find the exoskeleton, from which the set of the Voronoi edges was successively obtained after removal of certain pixels. As an ultimate step,
the various tiles were identified by using a component labeling technique. The proposed sequential algorithm for the construction of the discrete VD of objects in a digitized image can be conveniently implemented on standard sequential computers, as only five complete inspections of the picture are sufficient for the whole process. Moreover, all the computations can be performed on the array where the picture is initially stored.

Two parallel algorithms based on two different distance metrics have been reported in [77] and in [79] respectively: The first one is based on Euclidean distance function and proposes a two-dimensional cellular VLSI architecture and the second one is based on the d₄ distance metric using two-dimensional cellular automata (CA).

Sudha et al. [77] presented a parallel algorithm for computing the Voronoi diagram for collision-free path planning of robots. The algorithm constructs the Voronoi diagram on a binary image of the workspace based on the Euclidean distance function. The algorithm is made to enable VLSI implementation in a cellular architecture and such architecture has been described. Comparative study with other algorithms is also presented.

Tzionas et al. [79] proposed a new parallel algorithm for collision-free path planning of a diamond-shaped robot among arbitrarily shaped obstacles, which are represented as a discrete image, and its implementation in VLSI. The algorithm lays its foundation on a retraction of free space onto the Voronoi diagram, which is constructed through the time evolution of cellular automata, after an initial phase during which the boundaries of obstacles are identified and coded with respect to their orientation. The proposed algorithm
is both space and time efficient, since it does not require the modeling of objects or distance and intersection calculations. Additionally, the proposed two-dimensional multistate cellular automaton architecture achieves high frequency of operation and it is particularly suited for VLSI implementation due to its inherent parallelism, structural locality, regularity, and modularity.

Contributions [76, 80, 81] also had employed the Voronoi diagram for collision-free path planning. For instance, Rao’s work [76] encompasses learned navigation and terrain model acquisition problems. Specifically, he presented a framework that uses the same strategy to solve both problems. He employed the visibility graph as the underlying structure and the depth-first search as a searching strategy. He showed that any abstract graph structure that satisfies a set of four properties suffices as the underlying structure. He also showed that any graph exploration algorithm can serve as the searching strategy. As a result, he obtained other navigation structures to solve both problems. These methods provide paths that keep the robot as far from the obstacles as possible. In some cases, these methods are preferable to visibility graph methods that require the robot to navigate arbitrarily close to the obstacles, which is hard to implement if the robot motions are not precise.

The significance of his proposed framework is that it can be applied to any situation involving mobile robots or manipulators where a suitable navigation course could be found. In particular, applying Restricted Visibility Graph and Voronoi Diagram to circular robots is straightforward. Sugihara [80] also dealt with the approximation of generalized Voronoi diagrams by ordinary Voronoi diagrams.
Very recently, Lee and Choset [81] also followed Voronoi Diagram based approach in sensor-based exploration for convex bodies in order to form a new roadmap for a convex-shaped robot. They presented a new roadmap that can be used to guide a convex body to explore an unknown planar workspace, i.e., to map an unknown configuration space diffeomorphic to SE(2). This new roadmap is called the convex hierarchical generalized Voronoi graph (convex-HGVG). Since this roadmap is defined in terms of workspace distance information that is within line of sight of the convex body, we can use it to direct the robot to explore an unknown configuration space diffeomorphic to SE(2). The challenge in defining the roadmap is that SE(2), with holes removed from it, generally does not have a one-dimensional deformation retract. Therefore, they decomposed the punctured SE(2) into contractible regions, in which we define convex generalized Voronoi graphs (convex-GVG), and then connected these graphs with additional structures called convex-edges. We formally show that the convex-HGVG, which is the union of the convex-GVG edges and the convex-edges, is indeed a roadmap.

Now, we wish to give an overview of dynamic motion planning in different kinds of environments:

Rao et al. [82] presented an algorithm to navigate a point robot through a sequence of destination points amidst unknown stationary polygonal obstacles in a two dimensional terrain. The algorithm implements learning in the way of building a global terrain model by integrating the sensor information obtained during the course of navigation. This global model is used in planning the future navigational paths. This approach
prevents the robot from getting into localized detours, and also results in better navigational course, in an average case, compared to the algorithms without learning.

Buckley [83] had observed that the previous work pertaining to the motion planning problem for multiple translating robots in the plane assigned the robots priorities and generated motion plans for each robot based on the priority assigned through a combined space-time but did not mention how to assign priorities. In pursuit of overcoming this drawback, he showed that a careful priority assignment can greatly reduce the average running time of the planner and hence got motivated to introduce a new priority assignment method that attempts to maximize the number of robots which can move in a straight line from their start point to goal point, thereby minimizing the number of robots for which expensive collision-avoiding search is necessary. He observed that the method was extremely effective in sparse workspaces where the moving robots are the primary obstacles.

When multiple mobile robots are working in the same environment, planning of collision-free coordinated motion is inevitable. Liu et al. [84] proposed a feasible algorithm for planning such a motion of two mobile robots no matter how crude the constraints of obstacles are. In a multiple robots system, each robot must concern the motion of every other robot at least with regard to intersecting parts of their parts, if not non-intersecting parts of their paths without any risk of collision. They modeled this situation as Petri Net, which is a very useful model for describing and analyzing a system of concurrent events with constraints on the concurrence. In the Petri
Net, all of motion constraints are arranged as firing rules, and hence collision-
free coordination between the robots can be easily planned by manipulation
of the firing rules. Their proposed algorithm represents the environment as
quadtree and the robots are modeled as circles. The algorithm investigates
free regions recursively and arranges the robot's motion constraints in the
regions into the Petri Net until a collision-free path is determined or there is
no free region being explored. Therefore, the algorithm is capable of finding a
collision-free coordinated path of two robots always, if there exists one. It is
claimed that the algorithm works efficiently even in a complex environment
indebted to the generic property of geographical quadtree modeling for the
environment. However, it has to address many problems that emerge because
of the increase in the number of robots, when more than two robots are in the
environment.

Now, we discuss a case where obstacles are moving in stead of mobile
robots. Kehtarnavaz and Li [85] made an attempt toward establishing a more
realistic navigational environment by considering moving obstacles with
unknown trajectories. They framed a scheme for collision-free navigation that
constitutes an essential component of any autonomous vehicle. They predict
the future positions of obstacles using an autoregressive model and based on
the future positions obtain a collision-free path between the current and a
desired location of the vehicle. They also attempted to lay the foundation for
carrying out future research in modeling of navigational environments for
autonomous vehicles.
A work by Reif and Sharir [86] is also on motion planning in the presence of moving obstacles. They investigated the computational complexity of planning the motion of a body in 2-d or 3-d space with a view to avoiding collisions with moving obstacles of known, easily computed, trajectories. Prior to their work, there was no study on the computational complexities of dynamic movement problems which are of fundamental importance to robotics. They employed a unique method simulation of a Turing machine that uses time to encode configurations and established that the 3-d dynamic movement problem is intractable even if the body has a constant number of degrees of freedom of movement.

We emphasize here that an efficient and fast static motion planner is a vital ingredient of a dynamic motion planner in all cases discussed above, summarily, in partially known environments [82], in path planning of multiple robots with conflicting/common objectives, where robots of higher precedence are regarded as moving obstacles for robots of lower precedence ([83], [84]), and in time-varying environments, where robot motions have to be planned from the predicted motions of obstacles and replanning is needed if obstacle motion deviates from prediction ([85], [86]), a Voronoi planner of the type proposed in [79] is particularly required in the case where many motion planning problems, with different start and goal configurations within the same environment, are to be solved. In this case, we need to compute a complete spatial representation and search in this representation for paths connecting many pairs of start and goal configurations, at least once. For a marginally changing environment, an updating scheme can be used to avoid complete recomputation of the representation.
The visibility graph is an alternative approach, based also on the principle of retraction. In this method, a collection of lines is defined in free space such that it connects features of an object to those of another [87]. Asano et al. [87] gave a worst-case optimal algorithm for computing the visibility graph of line segments. This yields an $O(n^2)$-time algorithm for computing the Euclidean shortest paths between two points among polygonal obstacles with $n$ total edges. The visibility graph produces $O(n^2)$ edges for $O(n)$ features and, thus, it is a more complex approach than the algorithm proposed in [79] which produce only edges for the same number of features.

Employing potential functions that are developed for obstacle avoidance is another approach, known as, the potential field approach. It has a drawback due to local minima that trap the motion of the robot [88]. Also, the simple Newtonian potential is inadequate for deriving accurate representations of the object space. This necessitates the use of high-order potentials to obtain more accurate representations of space. In this case, the resulting path approaches the path computed using the Voronoi diagram, but it is associated with a much higher computational complexity [89].

Hwang and Ahuja [89] surveyed the work on gross-motion planning, including motion planners for point robots, rigid robots, and manipulators in stationary, time-varying, constrained, and movable-object environments. The general issues in motion planning are explained in the report. Recent approaches and their performances are briefly described, and possible future research directions are also discussed.
Methods based on mathematical programming have also been tried for path planning. These methods involve a set of inequalities for the representation of the problem space. However, as the number of inequalities increases, in order to obtain a more accurate description of space, these methods lead to complex nonlinear optimization problems [89].

Heuristic algorithms can also be suggested for path planning as done in [89]. For example, the algorithm proposed in [90] runs on the quadtree representation which is obtained from fast conversion of a real image obtained with a camera from the ceiling of the workspace. Since the real image expresses obstacle allocation in the workspace in good time, the algorithm can generate collision-free motions while following a change of obstacle allocation. Moreover, the quadtree controls all obstacle regions and other free regions in the workspace hierarchically in positioning, and hence the intermediate points and the closest points can be quickly selected out of the quadtree even if the shapes of the robot and the obstacles are complex. However, it uses the minimum width of the robot to eliminate narrow paths by computing closest distances to obstacles. This algorithm requires computations of distances and intersections and, thus, large execution times.

For motion planning of polygonal robots among polygonal obstacles, certain contributions like [91] have employed sweep volume methods. Zhu and Latombe [91] worked on constraint reformulation in a hierarchical path planner and developed algorithms by approximating the set of free configurations of the robot by a collection of rectangloid cells at successive levels of approximation and, at each level, searching the graph representing
the adjacency relation among these cells. Specifically, their algorithms address the main two computational issues underlying this approach: cell decomposition and graph searching. They implemented a planner which incorporates these algorithms and experimented with it on many examples also. These experiments show that their planner is significantly faster than previous planner based on the same general approach. They mainly focused on the planner’s cell decomposition algorithms. They used a constraint reformulation technique that consists of approximating the obstacles intersecting the cell to be decomposed by a collection of rectangloids and computing their complements in the cell. Two types of approximations are used, bounding and bounded approximations. This technique produces much less cells than previously proposed techniques, resulting in smaller search graphs.

The main disadvantage of these methods is that they require huge memory space capacity when compared with the algorithm proposed in [79], which requires minimal storage capacity, since each cell stores information only about its nearest neighbors.

Under the criterion of the shortest path, certain search methods such as depth-first, breadth-first, best-first, A*, bidirectional, and random searches have been proposed mainly for path-planning [89]. Algorithms for 2-d and 3-d path planning under the shortest path criterion and without any maximum clearance considerations, based also on CA architectures, have been proposed in [92] and [93]. For instance, Tzionas et al. [92] designed an algorithm for the calculation of the minimum cost path between a pair of points on a binary
map and its VLSI implementation using a new, multistate, 2-d cellular automata architecture. The main advantages of the proposed architecture are that storage requirements are reduced to a minimum and the speed of operation is very high.

Finally, the cell decomposition approach to path planning [94] partitions the free space into regions and identifies possible contacts (called critical contacts) between the robot and obstacles in each region. The algorithm proposed in [94] requires $O(n^5)$ time complexity for finding critical contacts. But, we find in the literature with less time complexity certain algorithms like [79].

The definition of various motion planning problems and approaches together with detailed explanations of landmark papers and a comprehensive list of references can be found in [74].

From above contributions, we find that the proposals presented in [77] and [79] have overcome the drawbacks associated with the methods employed in other proposals. These two proposals have employed a two-dimensional cellular architecture as the basis but differed on the metric used for determining distances. This architecture can be replaced by a single dimensional one when an LARPBS model is employed. Prompted by this idea, we have contributed two parallel algorithms based on two distance metrics on an LARPBS model.
2.6 Limitations of the literature

In this subsection, we bring out the limitations of the literature with regard the chosen problems and pinpoint how these limitations were turned into motivations for our proposals:

- There is a sequential algorithm for LCHP problem on a tree found in the literature. But, to the best of our knowledge, there is no parallel solution for the same.

- For the classes of some basic geometric problems mentioned in subsection 2.3, parallel solutions are found in the literature but not on the practical model, the LARPBS.

- Only a sequential algorithm is found for an advance problem, the rectilinear 3-centers, in the literature; no parallel solution for the same.

- Few parallel algorithms have been reported in the literature for robot path planning problem in Robotics. They are based on two-dimensional cellular architecture but none is found based on single dimensional VLSI architecture.