On Some Graph Layout Problems and Solution Strategies

1 Introduction

Many problems of practical interest can be represented by graphs. As a result, the development of algorithms to handle graphs is of major interest in computer science. Although many problems in graph theory are polynomial time solvable, there are an equally large number of problems that are NP-hard. NP stands for non-deterministic polynomial. NP-hardness means that we cannot find algorithms which find optimal solution for all instances of the problem in polynomial time [RM]. Two techniques are commonly in practice to handle with such problems, namely, Evolutionary Algorithms and Approximation Algorithms. Graph layout problems are one class of problems that are mostly NP-hard for general graphs. These problems have been approached using both evolutionary algorithms and approximation algorithms [JJM, RB]. In this thesis, it is proposed to find approximation algorithms for some Graph layout problems and exact results for special classes of graphs also.

Most of the available algorithms require direct access to the whole input. In the case of massive data sets, random access to the whole input is not possible since the data size is extremely large, it cannot be stored in random access memory (RAM). The solution to this problem is streaming in which data is presented as a stream [SLPA]. In this thesis, it is also proposed to deal with massive data sets for some Graph layout problems and maximum matching problem in graphs.

2 Evolutionary Algorithms

Evolutionary algorithms (EAs) are search methods that take their inspiration from natural selection and survival of the fittest in the biological world. EAs are principally a stochastic search and optimization method [KJ]. These algorithms differ from more traditional optimization techniques in that they involve a search from a population of solutions, not from a single point. Each iteration of an EA involves a competitive selection that weeds out poor solutions. The solutions with high fitness are recombined with other solutions by swapping parts of a solution with another. Solutions are also mutated by making a small change to a single element of the
solution. Recombination and mutation are used to generate new solutions that are biased towards regions of the space for which good solutions have already been seen [WPEA].

EAs are often viewed as a global optimization method although convergence to a global optimum is only guaranteed in a weak probabilistic sense. However, one of the strengths of EAs is that they perform well on noisy functions where there may be multiple local optima. EAs tend not to get stuck on a local minima and can often find a globally optimal solution. EAs are characterized by the representation of the individual, the evaluation function representing the fitness level of the individuals, and the population dynamics such as population size, variation operators, parent selection, reproduction and inheritance, survival competition method, etc.[KJ].

The structure of evolutionary algorithms is as follows: [ZM]

```
begin
  t ← 0
  initialize P(t)
  evaluate P(t)
  while (not termination condition) do
    begin
      t ← t+1
      select P(t) from P(t-1)
      alter P(t)
      evaluate P(t)
    end
end
```

The evolutionary algorithm is a probabilistic algorithm which maintains the population of individuals, \( P(t) = \{x^1_t, \ldots, x^n_t\} \) for iteration \( t \). Each individual represents the solution for the problem. Each solution \( x^i_t \) is evaluated to give some measure of fitness. Then a new population is formed (iteration \( t+1 \)) by selecting the more fit individuals (select step). Some members of the new population undergo transformation (alter step) by means of genetic operators to form new solutions. After some number of generations, the program converges and it is hoped that best individual represents near optimal solution [ZM].
Evolutionary algorithms perform well as approximating solutions to all types of problems because they ideally do not make any assumption about the underlying fitness landscape; this generality is shown by successes in fields as diverse as engineering, art, biology, economics, marketing, genetics, operations research, robotics, social sciences, physics, politics and chemistry [WPEA].

Compared to traditional optimization methods, such as calculus-based and enumerative strategies, EAs are robust, global, and may be applied generally without recourse to domain-specific heuristics, although performance is affected by these heuristics [KJ].

Evolutionary algorithms are widely used to solve stationary optimization problems. However, many real world problems are dynamic optimization problems. For dynamic optimization problems the fitness function, design variables, and/or environmental conditions may change over time [SY, JH, JB]. Evolutionary algorithm with proper enhancements works with dynamic (time-changing) problems [SY]. In dynamic environment, evolutionary algorithms are asked to track a moving optimum as closely as possible in time changing environment. Several approaches in evolutionary algorithms have been developed to address dynamic optimization problems such as maintaining diversity during the run via random immigrants, increasing diversity after a change, using memory schemes to store and reuse useful information. Among these approaches memory scheme is found to be beneficial for many dynamic optimization problems. There are two mechanisms to store useful information: implicit memory and explicit memory. For implicit memory schemes, evolutionary algorithms use genotype representation that contains redundant information to store good solutions to be used later. Explicit memory scheme use precise representations but split an extra memory space to explicitly store useful information such as good solutions and/or environmental information from the current generation for reuse in later generations or environments [SY].

In dynamic optimization we can identify two distinct phases. At first, the algorithm needs some time to search for the optimum. The period needed for this task is known as searching phase. Once the algorithm has found the optimum within a certain accuracy the algorithm must follow the moving optimum with a distance as small as possible. This phase is called a tracking phase [LS]. Evolution strategies in dynamic environment are also discussed in [LS].
3 Approximation Algorithms

In NP-hard problems, approximation algorithms relax the requirement of optimal solution. An approximation algorithm finds near optimal solution for all instances of the problem in polynomial time. In practice, it is usually hard to tell the difference between an optimal solution and near-optimal solution. There are two kinds of approximation algorithms: absolute approximation algorithms and relative approximation algorithms [RM].

3.1 Absolute Approximation Algorithms

An absolute approximation algorithm is a polynomial time approximation algorithm such that for some constant $k > 0$, $|A(I) - OPT(I)| \leq k$ for all input instances $I$ where $A(I)$ is the approximate result for the given problem for all input instances $I$ and $OPT(I)$ is the optimal result for the given problem for all input instances $I$. This is the best we can expect from an approximation algorithm for any NP-hard problem. But for some problems finding approximate solution within a given range is itself hard [RM].

3.2 Relative approximation Algorithms

In this strategy, quality of approximation algorithm is in terms of ratio between the value of its solution and that of optimal solution.

In case of minimization problem, performance ratio is

$$R_A(I) = \frac{A(I)}{OPT(I)}$$

and in the case of maximization problem, performance ratio is

$$R_A(I) = \frac{OPT(I)}{A(I)}$$

The ratio is always at least 1 and the algorithm produces a better approximation if the ratio is closer to 1 [RM].
3.3 Approximation Scheme

An approximation scheme for an optimization problem is an approximation algorithm that takes as input not only an instance of the problem, but also a value $\varepsilon > 0$ such that for any fixed $\varepsilon$, the scheme is a $(1+\varepsilon)$-approximation algorithm [TCRC]. An approximation scheme has the performance guarantee, $R_A(I, \varepsilon) \leq (1 + \varepsilon)$.

3.3.1 Polynomial-time Approximation Scheme (PTAS)

A polynomial approximation scheme is an approximation scheme $\{A_\varepsilon\}$ where each algorithm $A_\varepsilon$ runs in time polynomial in the length of the input instance $I$ [RM]. The running time of a polynomial-time approximation scheme can increase very rapidly as $\varepsilon$ decreases. For example, the running time of a polynomial-time approximation scheme might be $O(n^{2/\varepsilon})$ [TCRC].

3.3.2 Fully Polynomial-time Approximation Scheme (FPAS)

In PTAS running time of a polynomial-time approximation scheme can increase very rapidly as $\varepsilon$ decreases but ideally if $\varepsilon$ decreases by a constant factor, the running time to achieve the desired approximation should not increase by more than a constant factor [TCRC]. So the solution of this problem is FPAS. A fully polynomial approximation scheme is an approximation scheme $\{A_\varepsilon\}$ where each algorithm $A_\varepsilon$ runs in time polynomial in the length of the input instance $I$ and $1/\varepsilon$ [RM]. For example, the scheme might have a running time of $O((1/\varepsilon)^2 n^3)$. With such a scheme, any constant-factor decrease in $\varepsilon$ can be achieved with a corresponding constant-factor increase in the polynomial time.

4 New Class of Problems based on Graph Streams

4.1 Streaming

Streaming is an important model for computation on massive data sets. There are many applications of massive graphs in real world. For example, Web Graphs where nodes are web pages and the edges are link between pages, Call graphs where nodes correspond to telephone
numbers and edges to the calls between numbers that call each other during some time interval
[JSASJ]. Sale records collected everyday in supermarkets, etc. There are mainly two problems
that arise when computing with massive data sets. First one is the time complexity. The time
complexity of a computation is the amount of time required to complete that computation. It is a
function $f(n)$ of input size $n$. An algorithm is considered to be efficient when $f(n)$ is polynomial
of $n$. However, with extremely large input sizes, an algorithm with polynomial time complexity
whose polynomial contains terms of high orders becomes inefficient. The solution for this is
approximation algorithms. The second problem is storage capacity [JJ]. In the streaming model,
the entire graph cannot be stored in random access memory (RAM). Therefore, the algorithm has
no random access to the whole input which is required for most of the algorithms. The set of
edges has to be stored on an external device, like a disk or tape and is presented only as stream.
In this stream, each edge is presented exactly once, and each time the stream is read, edges may
appear in an arbitrary order. The stream can only be read as a whole and reading the whole
stream once is called a pass [SLPA].

In the streaming model, the computation is allowed to access the storage space in a sequential
fashion. Algorithms that make sequential access to the input data can be placed along a
spectrum. At one end of the spectrum, there are dynamic algorithms. These algorithms can use
memory large enough to hold the whole input. At the other extreme, there are streaming
algorithms that use only polylogarithmic space. Muthukrishnam proposed a middle approach,
where, for a graph with $n$ vertices, the algorithms use $n \cdot \text{polylog}(n)$ bits of space [JJ].

4.2 Problems in Data stream

There are many open problems in Data streams. One of those problems is maximum matching
problem. Maximum matching problem is to find a largest set of edges such that no two edges
share a vertex.

Following algorithms exist in the surveyed literature for bipartite graphs.

Algorithm in [JSASJ] finds approximate unweighted bipartite matching. This algorithm finds a
$2/3 - \varepsilon$ approximation of maximum matching in $O(\frac{\log 1/\varepsilon}{\varepsilon})$ passes. Algorithm in [SLPA] finds
approximate maximum matching in case of bipartite graphs. It starts with an arbitrary inclusion-
maximal matching $M$ and then repeatedly finds a set of vertex-disjoint $M$-augmenting paths and replace $M$ by an augmented version. This is a deterministic $1+\varepsilon$ approximation algorithm which requires only $O((1/\varepsilon)^5)$ passes over the input stream [SLPA].

For general graphs, McGregor [SLPA] proposed a randomized algorithm which finds a $1+\varepsilon$ approximation of a maximum matching in the semi-streaming model. It requires $\Omega(1/\varepsilon)^{\Omega(1/\varepsilon)}$ passes over the input stream in order to achieve the claimed success probability of $1-e^{-1}$. Even on the bipartite graphs it requires these many passes.

## 5 Graph Layout Problems

Graph Layout Problems are a particular class of optimization problems whose goal is to find a layout of an input in such a way that a certain objective is optimized. A linear layout is a labelling of the vertices of a graph with distinct integers. A large number of relevant problems in different domains can be formulated as graph layout problems. These include optimization of networks for parallel computer architectures, VLSI circuit design, information retrieval, numerical analysis, computational biology, graph theory, scheduling and archaeology [JJM]. Most interesting graph layout problems are NP-hard and their decisional versions are NP-complete, but, for most of their applications, feasible solutions with an almost optimal cost are sufficient [JJM]. These problems have been approached using both evolutionary algorithms and approximation algorithms. However, given the complexity of most problems approximation algorithms are still emerging and that too, in many cases, for special class of graphs only [JJM].

### 5.1 Definitions

A linear layout, of an undirected graph $G = (V, E)$ with $n = |V|$ vertices is a bijective function $\varphi: V \rightarrow [n] = \{1,\ldots,n\}$. A layout has also been called a linear ordering, a linear arrangement, a numbering or a labeling of the vertices of a graph. We denote by $\Phi(G)$ the set of all layouts of a graph $G$ [JJM].

Given a layout $\varphi$ of a graph $G = (V, E)$ and an integer $i$, we define the set $L(i, \varphi, G) = \{u \in V: \varphi(u) \leq i\}$ and the set $R(i, \varphi, G) = \{u \in V: \varphi(u) > i\}$. The edge cut at position $i$ of $\varphi$ is defined as
\[ \theta(i, \varphi, G) = |\{ uv \in E : u \in L(i, \varphi, G) \land v \in R(i, \varphi, G) \}| \]

and the \textit{modified edge cut} at position \( i \) of \( \varphi \) as

\[ \zeta(i, \varphi, G) = |\{ uv \in E : u \in L(i, \varphi, G) \land v \in R(i, \varphi, G) \land \varphi(u) \neq i \}|. \]

The \textit{vertex cut} or separation at position \( i \) of \( \varphi \) is defined as

\[ \delta(i, \varphi, G) = |\{ u \in L(i, \varphi, G) : \exists v \in R(i, \varphi, G) : uv \in E \}|. \]

Given a layout \( \varphi \) of \( G \) and an edge \( uv \in E \), the \textit{length} of \( uv \) on \( \varphi \) is

\[ \lambda(uv, \varphi, G) = |\varphi(u) - \varphi(v)|. \]

\textbf{5.2 Some Graph Layout Problems}

\textbf{Bandwidth (BW) Minimization Problem}: Given a graph \( G = (V, E) \), find a layout \( \varphi^* \in \Phi(G) \) such that \( BW(\varphi^*, G) = \text{MINBW}(G) \) where

\[ BW(\varphi, G) = \max_{uv \in E} \lambda(uv, \varphi, G). \]

\textbf{Minimum Linear Arrangement (LA)}: Given a graph \( G = (V, E) \), find a layout \( \varphi^* \in \Phi(G) \) such that \( LA(\varphi^*, G) = \text{MINLA}(G) \) where

\[ LA(\varphi, G) = \sum_{uv \in E} \lambda(uv, \varphi, G). \]

\textbf{Cutwidth (CW) Minimization Problem}: Given a graph \( G = (V, E) \), find a layout \( \varphi^* \in \Phi(G) \) such that \( CW(\varphi^*, G) = \text{MINCW}(G) \) where

\[ CW(\varphi, G) = \max_{i \subseteq [V]} \theta(i, \varphi, G). \]

\textbf{Modified Cut (MC) Minimization Problem}: Given a graph \( G = (V, E) \), find a layout \( \varphi^* \in \Phi(G) \) such that \( MC(\varphi^*, G) = \text{MINMC}(G) \) where

\[ MC(\varphi, G) = \sum_{i \subseteq [V]} \zeta(i, \varphi, G). \]

\textbf{Vertex Separation (VS) or Pathwidth (PW) Minimization Problem}: Given a graph \( G = (V, E) \), find a layout \( \varphi^* \in \Phi(G) \) such that \( VS(\varphi^*, G) = \text{MINVS}(G) \) where

\[ VS(\varphi, G) = \max_{i \subseteq [V]} \delta(i, \varphi, G). \]
**Edge Bisection (EB) Minimization Problem:** Given a graph $G = (V, E)$, find a layout $\varphi^* \in \Phi(G)$ such that $EB(\varphi^*, G) = \text{MINEB}(G)$ where $EB(\varphi, G) = \delta \left( \left\lfloor \frac{1}{2} |V| \right\rfloor, \varphi, G \right)$.

**Antibandwidth (ABW) Maximization Problem:** Given a graph $G = (V, E)$, find a layout $\varphi^* \in \Phi(G)$ such that $ABW(\varphi^*, G) = \text{MAXABW}(G)$ where $ABW(\varphi, G) = \min_{uv \in E} \lambda(uv, \varphi, G)$.

**K-Page Crossing Number Minimization Problem:** In this problem, the vertices of a graph are placed along the spine of the book and every edge is completely drawn in one of the $K$ pages. The minimum number of edge crossings over all $K$-page book drawings of a graph $G$ is called $K$-page book crossing number of $G$.

**Page Number Minimization Problem:** A book embedding of a graph $G$ comprises embedding the graphs nodes along the spine of a book and embedding the edges on the pages so that the edges embedded on the same page do not intersect. The pagenumber of a graph is the thickness of the smallest (in number of pages) book into which $G$ can be embedded.

There are some classes of graphs which are optimally solvable in polynomial time for these problems [JJM].

5.2 **Short literature review with focus on evolutionary, approximation and streaming for graph layout problems in general**

Bandwidth minimization problem is NP-complete for general graphs. This problem remains NP-complete even for certain restricted classes of graphs. Bandwidth minimization problem is approached through approximation algorithms and evolutionary algorithms [ABF, ABRF, AJBF, AJF, ALBF]. NP-completeness of this problem is shown for trees with maximum degree 3, caterpillars with hair-length at most 3 and for caterpillars with at most one unbounded hair attached to the backbone [JJM]. Even for cyclic caterpillars with hair length 1, bandwidth minimization problem is NP-complete [JP]. For some classes of graphs, bandwidth minimization problem is optimally solvable in polynomial time. For caterpillars with hair-length $\leq 2$, hypercubes, butterflies, interval graphs, chain graphs, complete k-level t-ary tree, square grids [JJM], rectangular grids, cubic grids, toroidal meshes, unit interval graphs, 3-dimensional grids [JP] this problem is optimally solvable. Approximate results are also available for certain classes
of graphs such as $\delta$-dense graphs, asteroidal triple free graphs, caterpillars, GHB-tree. For general graphs, bandwidth has several polylogarithmic approximation algorithms running in polynomial randomized time. Bandwidth does not belong to PTAS [JJM]. In fact, for any constant $k$, it is NP-complete to find any $k$-approximation even for caterpillars [JP].

Cutwidth minimization problem is NP-complete in general. Its NP-completeness is shown for some classes of graphs also e.g., graphs with maximum degree 3, planar graphs with maximum degree 3, grid graphs, unit disk graphs [JJM], split graphs and chordal graphs [JP]. Cutwidth minimization problem is optimally solvable in polynomial time for some classes of graphs. It is optimally solvable for trees, hypercubes, d-dimensional c-ary cliques, max degree $\leq \Delta$ and treewidth $\leq k$, ordinary 2- and 3-dimensional meshes, toroidal and cylindrical 2-dimensional meshes, complete p-partite graphs [JJM], complete binary trees, abelian cayley graphs, de Bruijn graphs, bounded degree partial w-tree, threshold graphs, bipartite permutation graphs, unit interval graphs [JP] in polynomial time. An algorithm is proposed by Yannakakis [MY] which computes the planar cutwidth of the tree correctly in linear time There is a PTAS for dense graphs whose time complexity is $O(n^{1/e^2})$. An $O(\log^2 n)$-approximation for general graphs with time complexity $\text{pol}(n)$ is reported in [JJM]. Recently, a scatter search method which finds approximate solution is proposed for this problem [JRAE]. A branch and bound algorithm is also proposed which solves all the small-sized instances (up to 32 vertices) as well as some of the large-sized instances tested (up to 158 vertices) [RJAE].

Minimum linear arrangement problem is NP-complete in general [JP]. Minimum linear arrangement problem is optimally solvable for some classes of graphs such as trees, rooted trees, rectangular grids, square grids, 2-dimensional cylinder, hypercubes, de Bruijn graphs of order 4, d-dimensional c-ary cliques, complete p-partite graphs [JJM], proper interval graphs, certain halin graphs, outerplanar graphs, unit interval graphs, chordal graphs [JP] in polynomial time. There are approximate results also for minimum linear arrangement problem. For general graphs there is a $O(\sqrt{\log n \log \log n})$-approximation whose time complexity is $\text{pol}(n)$ [JP]. For planar graphs and interval graphs also there are approximation algorithms. There is a PTAS for dense graphs whose time complexity is $n^{O(1/e^3)}$ [JJM]. An EA is presented for this problem with
knowledge-based operator designed to search the solution space [RS]. A memetic algorithm is also presented to compute near optimal solutions for the MINLA problem [EJJ].

Vertex Separation or Pathwidth minimization problem is NP-complete in general. Its NP-completeness is shown for planar graphs with maximum degree 3, bipartite graphs, grid graphs and unit disk graphs [JJM], starlike graphs, chordal graphs, split graphs [JP]. Pathwidth minimization problem is optimally solvable in polynomial time for some classes of graphs such as trees, cographs, permutation graphs, n-dimensional grids [JJM], bounded degree partial w-tree, unicyclic graphs, 2-dimensional grids, cylinders, tori, 3-dimensional grids, k-starlike, split, primitive starlike graphs, trapezoid, permutation graphs, biconvex bipartite graphs, circular-arc graphs, cographs, comparability graphs of interval orders [JP]. There are approximate results also for pathwidth minimization problem. For general graphs there is $O(\log^{1.5} n)$-approximation whose time complexity is pol(n). There are approximate results for planar graphs [JJM], outerplanar graphs and halin graphs also [JP]. In the surveyed literature there are no evolutionary algorithms for this problem.

Edge Bisection Minimization problem is NP-complete in general. This problem is NP-complete for graphs with maximum degree 3 and for graphs with bounded maximum degree. This problem is also NP-complete for d-regular graphs [JJM]. It is optimally solvable in polynomial time for some classes of graphs such as trees, hypercubes, d-dimensional c-ary arrays, ordinary 2- and 3-dimensional meshes, toroidal and cylindrical 2-dimensional meshes, toroidal 3-dimensional meshes, grid graphs, treewidth $\leq k$, cube-connected cycles graphs [JJM]. There is a PTAS for general graphs whose time complexity is $f(n, \varepsilon)$. For general graphs there is a $O(\log^{1.5} n)$-approximation whose time complexity is pol(n) and $O(\log n)$-approximation for planar graphs whose time complexity is pol(n) [JJM]. A hybrid evolutionary algorithm for the edge bisection problem of the graphs is proposed in [RS].

Exact page number results for some classes of graphs exist in literature. Optimal 3-page book embedding of the FFT, Benes and Barrel Shifter Networks are presented in [RG]. Every $n$-vertex $d$-valent graph can be embedded using $O(dn^{1/2})$ and for every $d > 2$ and all large $n$, there are $n$-vertex $d$-valent graphs whose pagernumber is at least $\Omega \left( \frac{n^{1/2 - 1/d}}{\log^2 n} \right)$ [FFA]. 5-page embedding of cayley trivalent graphs is presented in [YY]. A genetic algorithm for finding the page number of
interconnection networks is also proposed [NMIA]. A hybrid evolutionary algorithm for page number minimization problem is proposed in [DKG].

A polynomial time approximation algorithm is proposed to generate a drawing of graph with $O(m^2/k^2)$ crossings on $k$-pages [FLO]. FLCNP is similar to 2-page book crossing number problem. In this problem, the vertices of graph are placed in a fixed order along a horizontal line and the objective is to minimize the number of edge crossings. Some heuristics for fixed linear crossing number problem (FLCP) are discussed and their worst case performances are analyzed in [RC]. A randomized polynomial time $0.878 + 0.122\rho$ approximation algorithm for FLCNP, where $\rho$ is the ratio of the number of conflicted pairs to the crossing number is presented in [RCB]. A genetic algorithm and an evolutionary algorithm for the 2-page book drawing problem of graphs is also proposed in [HOE] and [RB] respectively. Hongmei [HOAE] also gave the parallelization of genetic algorithms for this problem. Besides, Neural networks have also been employed for 2-page crossing number problem [HSM]. Chung [FFA] derived exact results of X-trees and 2 dimensional meshes; Hongmei [HAE] solved it for a class of circulant graphs.

Given a graph $G(V,E)$ it is NP-complete to decide whether $ab(G) \geq 2$ where $ab(G)$ stands for antibandwidth of graph $G$ [LT]. Lower and Upper bounds for some classes of graphs are also discussed in [LT]. For antibandwidth maximization problem exact results are obtained for certain classes of forests and asymptotically lower bounds for grids and hypercubes [ZD]. A labeling algorithm and tight upper bounds for Hamming graphs is also given in [SRDLI]. Heuristic for constructing lower bound for antibandwith parameter of arbitrary bipartite graphs is given in [PP]. GRASP with path relinking heuristics for the antibandwidth problem is proposed by Duarte et al. [LT]. A memetic algorithm for the antibandwidth maximization problem is proposed in [RB].

In the literature surveyed so far, there are no streaming algorithms for graph layout problems.

### 5.3 Survey for selected problems

#### 5.3.1 Cyclic Cutwidth Minimization Problem

In linear embedding, a region is defined as the area between two adjacent vertices of a graph. The cut of the region is the number of edges that cross the region from the left or right. In the
cyclic embedding of graph, vertices are embedded onto a cycle. Any edge that connects vertices in graph will also connect vertices in the cyclic embedding of the graph. A region between two adjacent vertices in a cyclic embedding is the triangular area created by two adjacent vertices and the center of the circle. The cut of a region in a cyclic embedding is the number of edges that cross over the given region [MJ].

The maximum cut of a particular embedding of a graph is the largest cut that occurs on the graph. The cutwidth of the graph is the minimum of all possible maximum cuts over all possible embeddings [MJ]. Many different people have proved cyclic cutwidth for some classes of graphs. It has also been proved that there is no single solution to the cutwidth problem of a general graph. Rios [MJ] developed a formula for cyclic cutwidth of a complete graph. Schroder [MJ] has given the cyclic cutwidth of a two-dimensional mesh \( P_m \times P_n \). Sciortino et al., [VJR] has proved the cyclic cutwidth for three-dimensional mesh \( P_2 \times P_2 \times P_n \). Chavez and Trapp [MJ] have proved that in the case of trees cyclic cutwidth and linear cutwidth are equal. Johnson [MJ] has given partial solution to the cyclic cutwidth of the complete bipartite graph. Allmond [HM] has proved the lower bounds and exact results for cyclic cutwidth of complete \( n \)-partite graph and upper bound for complete tripartite graph.

In the surveyed literature, there are neither approximation algorithms nor evolutionary algorithms available for this problem.

### 5.3.2 Cyclic Bandwidth Minimization Problem

Given a graph \( G \) with \( n \) vertices and a bijection \( f : V(G) \to \{1,2,...,n\} \), the cyclic bandwidth of graph is defined by \( B_C(G) = \min_{f \in F} B_C(G,f) \) where \( F \) is the set of possible bijections of \( G \),

\[
B_C(G,F) = \max_{u,v \in E(G)} |f(u) - f(v)|_C \text{ and } |x|_C = \min\{|x|, n-|x|\}.
\]

Lam et al. proved that bandwidth \( (G) \leq 2 B_C(G) \). It means that, if there is an \( \alpha \)-approximation algorithm for cyclic bandwidth minimization problem then there is \( 2\alpha \)-approximation algorithm for bandwidth minimization problem. There are also some graphs for which bandwidth and cyclic bandwidth are same. Lam et al. showed that trees and rectangular grids are examples of such graphs [EMR]. In [WPW] it is given that the maximum cyclic bandwidth of graphs of order

13
p with adding an edge $e \in E(\overline{G})$ is $2B_C(G)$ if $B_C(G) \leq \frac{p}{8}$ and it is $\left[ \frac{1}{3} \left( \frac{p}{2} + 2B_C(G) \right) \right]$ if $B_C(G) > \frac{p}{8}$.

In the surveyed literature, there are neither approximation algorithms nor evolutionary algorithms available for this problem.

### 5.3.3 Cyclic Bandwidth Sum minimization problem

Given a graph $G$ with $n$ vertices cyclic bandwidth sum of $G$ with respect to a layout $f$ is defined by $CBS_f(G) = \sum_{u,v \in E(G)} | f(v) - f(u) |$, where $|x|_p = \min\{|x|, p-|x|\}$. Cyclic Bandwidth Sum of $G$ is defined by $CBS(G) = \min\{CBS_f(G) : f$ is a proper labeling of $G\}$. In [YJ] Ying-Da Chen and Jing-Ho Yan have given a necessary and sufficient condition for $BS(G) = CBS(G)$. They have also found cyclic bandwidth sum of complete bipartite graph. Recently, variable neighbourhood search and scatter search algorithms are proposed for this problem in [DKGa, DKGb].

In the surveyed literature, there are no approximation algorithms available for this problem.

### 5.3.4 Cyclic Antibandwidth maximization problem

The cyclic antibandwidth maximization problem is to embed an $n$-vertex graph into the cycle such that the minimum distance (measured in the cycle) of the adjacent vertices is maximized. This problem is NP-hard. Exact results for meshes, tori and asymptotics are proved in [OLI]. For the cyclic antibandwidth of $d$-dimensional hamming graphs equality of antibandwidth maximization problem and cyclic antibandwidth maximization problem is proved in [SRDLI]. A memetic algorithm for cyclic antibandwidth maximization problem is also proposed in [RB].

In the surveyed literature, there are no approximation algorithms available for this problem.

### 6 Objectives

1. To explore and identify specific applications of evolutionary algorithms to graph layout problems among cyclic cutwidth, cyclic bandwidth, cyclic bandwidth sum and cyclic antibandwidth problems.
2. To explore, identify and propose a dynamic evolutionary strategy for a layout problem in a graph streaming situation. An approximate algorithm for the same problem will also be explored.
3. To explore approximate algorithms for a specific class of graphs for a layout problem.
4. To explore exact results for some layout problems.

References


